



TRANSITIVE[®]

I|C|E[™]
systems

Migrating Applications from Solaris[™] /SPARC[®] to Industry-Standard Platforms

“The QuickTransit[®] technology from Transitive is a truly breakthrough achievement. Not only does it provide a solution to many hitherto intractable software migration problems, but it opens up a new world of exciting possibilities for rethinking the traditional relationship between hardware and software design.”

– Dr. Geoffrey Robinson
MacRobert Award Committee and
former Director of Technology, IBM UK

Introduction	2
QuickTransit [®] Technology	3
Migration Overview	4
Migration Methodology	5
Customer Examples	10
Conclusion	11
Next Steps	11

Introduction

Migration from one platform to another is traditionally considered a difficult and costly project. For this reason many companies keep their old systems, both hardware and software, running for as long as possible, and in some cases well past the support deadlines provided by their suppliers. Often the upgrade paths to more powerful and efficient newer generations of the legacy hardware platform are blocked by operating system (OS) compatibility issues. These issues will prevent users from providing additional computing resources to cope with increasing demands on performance and efficiency. Even if customers are able to put up with those constraints, the question should be asked; "But what happens when the old system fails?" When the usual sources of parts and service are no longer available, desperate companies will purchase replacement parts for their systems on eBay or other unlikely sources. The logical path involves migrating the system and/or applications to the newest version, or at least one that will be supported for a while longer. There are two approaches to this kind of migration.

A system migration would involve picking up the entire software stack, from the applications all the way down to the operating system, and moving it on to new hardware. Hardware virtualization products may provide physical-to-virtual (P-to-V) tools that make this move possible, if not simple. However, this kind of system migration will only work when the hardware architecture is identical between the old and the new systems, for instance from older x86 servers to the newest x86 servers.

If, however, you want to move to a different hardware architecture, for instance from SPARC[®] to x86-64, then a different approach, that of application migration, is required because of the difference in the hardware instruction sets. Additional complexity ensues when you change operating systems, for instance from Solaris[™] to Linux[®]. An application migration between disparate hardware/software platforms requires each application to be considered independently for the best migration option:

- Upgrading to the latest release of a commercially available application
- Recompiling or porting custom applications to the new platform.
- Running the existing application unchanged using QuickTransit[®] on the new platform

The objective of this paper is to address the different approaches that may be used for migrating applications from one hardware/software platform to another. Specifically, it will address the migration from Solaris/SPARC systems to industry-standard platforms.

Transitive[®], the leader in cross-platform virtualization, addresses this need with its product, QuickTransit, which allows applications compiled for one OS and hardware architecture to run in another without recompilation. While effective in doing this with high performance and low effort, QuickTransit must be considered to be one part of a complete migration strategy as it addresses migration of individual applications and is not a system migration solution.

QuickTransit is a dynamic binary translator that operates on Solaris/SPARC applications at execution time and effectively converts the application to operate in the new environment. QuickTransit is, in actuality, three products:

“Desperate companies will purchase replacement parts for their systems on eBay or other unlikely sources.”



- QuickTransit® for Solaris™/SPARC®-to-Linux®/x86-64
- QuickTransit® for Solaris™/SPARC®-to-Solaris™/x86-64
- QuickTransit® for Solaris™/SPARC®-to-Linux®/Itanium®

When approaching a migration project, especially from one system architecture to another, the applications will provide the largest challenge. Making them work properly in the new environment is the goal. As a general rule, a natively compiled application will run faster than a translated application, so this is a consideration when migrating applications from Solaris/SPARC platforms to newer versions of Solaris or to other platforms. A corollary to that rule is that a Solaris/SPARC application will typically run faster translated on a new industry-standard system than it will as a native application on older SPARC hardware. Therefore, while QuickTransit is an effective way to move an application workload from a Solaris/SPARC system to other platforms it is not the only way to perform this migration. Determining how this migration will be completed and how applications will be carried forward is the purpose of this project. Each of these possible activities will be discussed in this paper.

QuickTransit Technology

QuickTransit is Transitive's product for cross-platform virtualization. It allows applications that were written for one operating system and hardware combination to run in a different operating system and hardware combination without making any changes to the source or binary code. There is no need to recompile or port that application. QuickTransit makes the existing binaries run by employing dynamic binary translation technology.

As the application attempts to execute, the execution path is grouped into a set of instructions that will always be executed in sequence called a *basic block*. QuickTransit decodes the block, understands what function it is trying to perform, optimizes that function and then generates the equivalent code for the new environment. The generated code is then cached so that the next time the program executes that same path it does not have to be translated again. Operating system calls are mapped to the new operating system's specifications and returns from the OS calls are mapped back to the results expected by the original program. Since this is a dynamic process, only those instructions and paths that are being executed are actually translated. As new paths are executed they are translated until there is a working set that encompasses the needs of the application at that time. As the application continues to execute, QuickTransit observes its behavior and further optimizes those basic blocks that are more frequently executed into larger blocks based on the application's behavior and thus further increases the run-time performance. Through this process the application runs at near native speed on the new platform.

When running an application translated by QuickTransit, the performance you get may vary based on the characteristics of your workload. There are a few qualifiers that should be noted. If your application is coming from an older Solaris/SPARC system, perhaps one running a Sun™ UltraSPARC® II+ processor (460 Mhz), the performance of a new industry-standard processor is many times faster. Allowing for the translation process, tests have shown that applications running translated on modern x86 processors will generally perform two to five times faster than on their original platform, as shown in Figure 1. Even compared to the most recent Sun UltraSPARC processor generation

“QuickTransit is part of a complete migration strategy.”

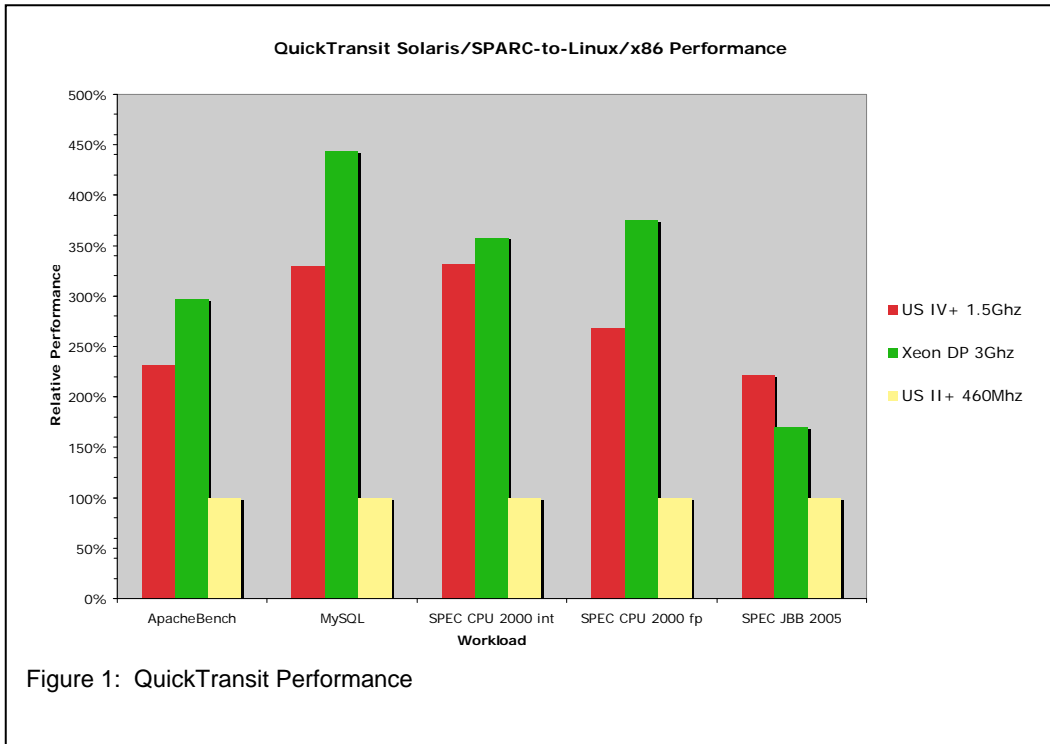


Figure 1: QuickTransit Performance

(IV+), application binaries will typically run slightly faster on the latest generation of Intel® and AMD processors with QuickTransit.

There are a few limitations on applications that are effectively translated by QuickTransit:

- Since there is no Solaris/SPARC kernel running in the new environment, the application can have no dependencies on any Solaris/SPARC kernel modules.
- If the application is dependent upon specific peripherals requiring special Solaris kernel device drivers, then it is not a good candidate for translation.

Migration Overview

Migration of applications from one platform to another is a difficult and time-consuming process, and will sometimes carry with it a hefty price tag. Justifying the cost of application porting in a time of declining IT budgets is very difficult. Studies have shown that a project to port an application from one platform to another, without making any functional upgrades to the operation of the application, can cost as much as 40% of the original cost of developing the application.

Moving an application from one version of an operating system to another requires specific knowledge of the differences between the old and new operating environments. Changes to OS calls may be complicated by differences in the parameters, changes to the return values, or even incompatibility with the new version of the OS. Even with commercial applications, moving from one version to another may involve many changes to the configuration parameters and ways in which the application can be used. Internally-developed applications or applications written on contract by an ISV to a customer's specifications (also known as "bespoke" applications) will require knowledge of the internal workings of the application to be moved from one environment to another. In some cases, the source code for the bespoke application has been lost, leaving the application trapped on an "orphaned" platform with no recourse but to continue running it there.

“Even compared to the most recent Sun processor generation, application binaries will typically run faster on the latest generation of Intel processors with QuickTransit.”



Some applications have other applications embedded within them. For instance, an application may collect data and provide reports to users through the use of a database management system that is completely within the application and hidden from the users' view. If that embedded application can't easily be upgraded or moved to another platform, it creates another environment for a trapped application.

The options available to the migration project are:

- To get other commercial applications for the new platform
- To recompile internally-developed applications to run on the new platform
- To run the applications translated with QuickTransit on the new platform
- Abandon the applications, because they will no longer be needed on the new platform

Knowing how to approach the project, which applications fall into each category and how to address each for the optimum use of resources – both during and after the project – is the key to project success. This requires knowledge, experience and a repeatable methodology.

Migration Methodology

Determining the best approach to migrating application workloads from Solaris/SPARC to other platforms involves careful consideration of each application. Applications will generally fall into one of the following categories:

- A. A commercial application created by an ISV or open source application
- B. An application developed internally by your company for your own proprietary needs or created by an ISV under commission for your company
- C. A multi-tier application that involves both a commercial and an internally developed application which work together as a single unit
- D. A special case of commercial or internally developed applications are those mission-critical applications that are almost entirely based on the Java™ programming language and environment (NOTE: Java-based installers, configuration or admin tools are not included in this grouping as they are run infrequently.)

In each category there can be several approaches that may be used in a migration scenario, and each of them will be addressed in the sections that follow. This paper is intended to be a high-level look at the process, but will identify specific actions that may be appropriate under certain circumstances.

“Successful migration projects require knowledge, experience and a repeatable methodology.”

Case A: ISV Applications

As a general rule, if a commercial application has a Linux/x86 equivalent, then you should use that version of the application, if possible. The ease of migrating from the Solaris/SPARC version to the Linux/x86 version will depend on the availability of compatible versions. If the application on both sides is the same version, then it may be a fairly straightforward move from one to the

other requiring no significant changes to the configuration files, input or data formats. If there is a version difference between the two, there may be some extensive changes required in order to make the application work in the same way. The effort required to upgrade to the new version may be very small, and thus worth doing.

However, if the upgrade effort is very high then other options, like dynamic binary translation, should be considered. In some cases, like that of the Oracle® 8 Database, there isn't an equivalent version available for Linux/x86. In this situation the effort to upgrade to Oracle 10 or 11 can be very large. Therefore, the recommended approach for Oracle 8 would be to treat it the same as a custom application and run it translated with QuickTransit as in Case B, below.

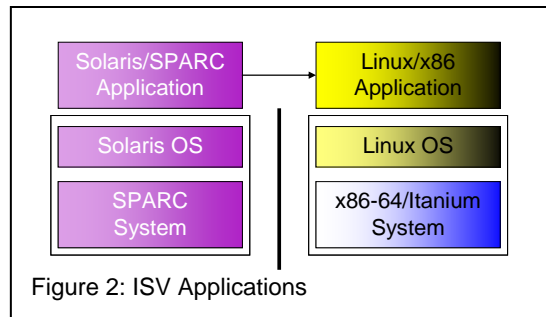
Case B: Custom Applications

Migrating a custom application from a Solaris/SPARC system to an industry-standard x86 system with either the Solaris or Linux operating system is not always a straightforward process, and may encounter one or more of these obstacles:

- The only version of the Solaris OS available on x86 systems is Solaris 10. Given the differences between Solaris 10 and earlier versions of the OS, effort will have to be invested to make the application compatible with the newer version.
- If you are moving to a Linux/x86 system, the application will have to be changed to accommodate the differences between the Solaris and Linux OS services and application programming interfaces (APIs).
- Industry-standard x86 architecture orders the bytes stored in memory in a different order than the SPARC architecture – known as endianness. Getting the endianness wrong in a program may lead to confusion and very esoteric bugs that will be difficult to correct. Assumptions made about endianness in a program may lead to optimizations in the code that don't convert properly when porting an application.

The traditional approach for porting an application from one environment to another requires that the program be recompiled and then adapted for OS calls and endian issues. Projects such as these tend to overrun both their budgets and schedules for a number of reasons:

- The project requires specialized knowledge of both the Solaris/SPARC environment as well as the target environment. Such combined knowledge may not be available within the application development organization and may have to be obtained from outside (i.e. a contractor or migration specialist)



“Porting an application requires specialized knowledge of both the old and new environments that is not necessarily available within a typical IT shop.”

- The original application development team may not be available for this porting project. This may include those with specialized knowledge of proprietary assumptions that were used in the development of the original application. Missing this knowledge typically results in substantially longer debug cycles.
- The source code for the application may no longer be available, or may not match the deployed version of the application. In any event, this would make porting impossible. Redevelopment of the application would inevitably drive up the cost.
- Any good porting plan will include extensive testing to ensure that the new version behaves the same as the original. Issues like endianness may create situations in which many cycles through the development/test cycle are required to find and eliminate problems on the new system.

As mentioned before, the cost of porting an application without making any changes to its functionality may be as much as 40% of the original cost of the application.

An alternative to this kind of costly porting project is to consider running this type of application translated with QuickTransit. As described above, QuickTransit dynamically generates and optimizes native code that is equivalent to the original Solaris/SPARC code. As the application continues to execute and the most frequently used paths are identified, they are further optimized to get the best possible performance from the translated application.

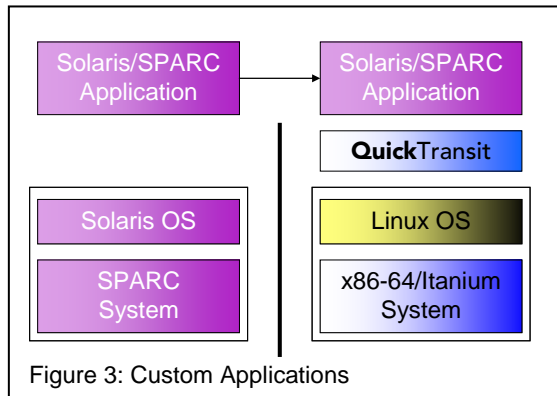


Figure 3: Custom Applications

If the custom application is still actively developed and maximum performance is required, then a port of the application should be considered. As application porting can take anywhere from several months to more than a year, depending on the availability of the required expertise for the project, it may be appropriate to run the application translated with QuickTransit while the porting is proceeding. In that way, enterprises can take advantage of the savings in energy use and server floor space and apply those savings to the porting effort. Additional potential benefits from this approach are a happier user community and less pressure on the porting team. Either way, QuickTransit is recommended in these situations as it will provide nearly all of the performance advantages without any of the porting-associated issues.

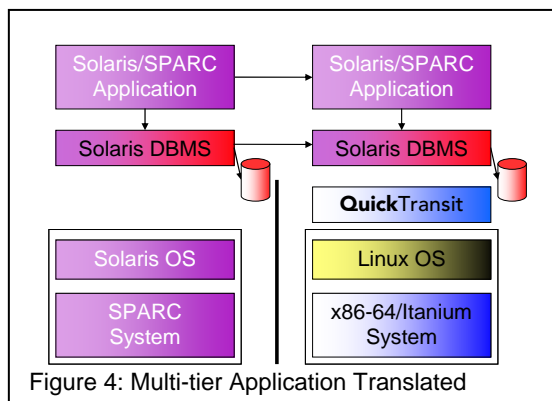


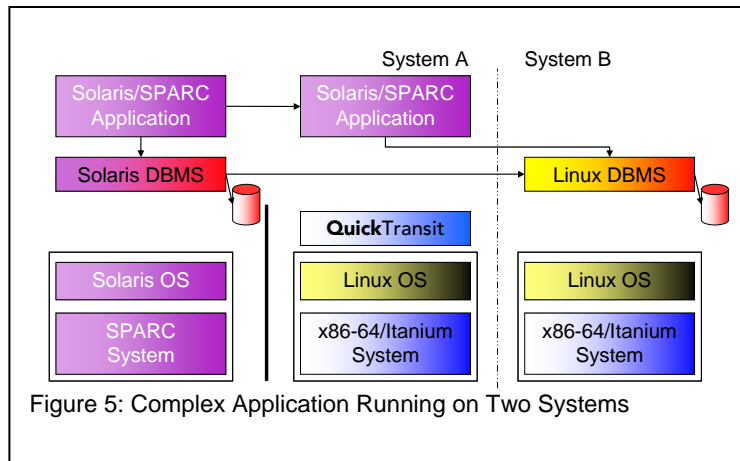
Figure 4: Multi-tier Application Translated

“It may be appropriate to run the application translated with QuickTransit while the porting project is proceeding.”

Other applications may work very well translated with QuickTransit, specifically those ISV applications without an equivalent in the target system. In the case of Oracle 8, which has no equivalent Linux/x86 version, running the application translated is a viable, and indeed, a preferred option. Tests have shown that Oracle 8 performs well as a translated application.

Case C: Multi-Tier Applications

Many systems/solutions are made up of more than one component, and they are often of different types. For instance, a custom application may use an ISV database management system (DBMS) for access to volumes of data, or it may be a custom application that is tied directly to an application server or web server. In this case, each piece of the multi-tier application should be considered individually – the custom portion of the application should be treated as in Case B and the ISV application should be treated as in Case A above. We recommend that you make the best



choice for each portion individually and then be sure that it is the best choice for the combined applications. Either portion may be run translated with QuickTransit or natively and still communicate in the same way as before. If the application is too difficult to decompose into its component parts, then running the entire multi-tier application translated (i.e., treating it as a variant of Case B in Figure 3) is an excellent alternative.

If the ISV portion of the multi-tier application is easily upgraded to an equivalent version then it should be run in native mode on the new platform. On the other hand, as in the case of ISV applications like Oracle 8, mentioned previously, where an equivalent is not available, then running translated with QuickTransit will provide savings and performance advantages.

In addition, it may be possible to run the two portions of the multi-tier application in more than one system or more than one virtual machine, assuming that the communication mechanism used between the two portions of the multi-tier application allow it. In this way the portions of the application may be positioned within the IT infrastructure of the enterprise and provide additional flexibility to the system administrators.

“We recommend that you make the best choice for each portion of the application individually and then be sure that it is the best choice for the combined applications.”

Case D: Java Applications

Java was introduced by Sun Microsystems in 1996 with the promise of portability that was characterized as *Write Once, Run Anywhere™*. The objective of Java was to provide the same object-oriented programming interface regardless of the architecture of the machine that it is running on. The constructs of Java are similar to C++ and other object-oriented languages, and depend on a Java Virtual Machine (JVM™) being available on every platform where you want to run the application. These JVMs are provided by a variety of companies and are supposed to conform to the Java “standards” that Sun and its community process control. Over the past decade, Java has been well accepted and many new applications have been created in this language. A Java has matured some differences between JVMs have surfaced that prevent full portability across platforms. Some are based on the requirement to execute native platform functionality through a feature known as the Java Native Interface (JNI™), and others are just differences that occur when multiple teams work on the same project.

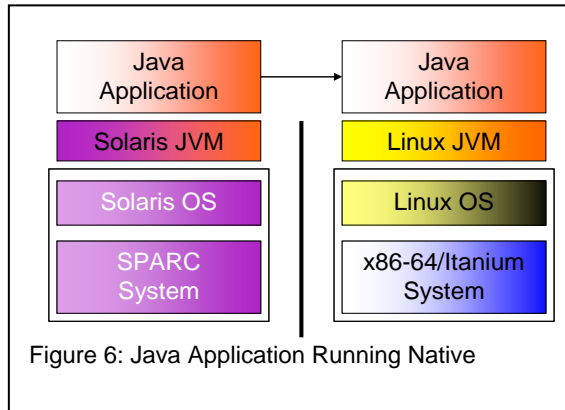


Figure 6: Java Application Running Native

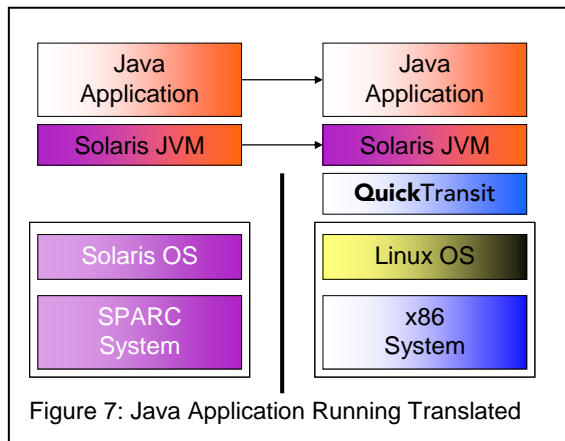


Figure 7: Java Application Running Translated

Whenever possible, Java programs should be run on a native JVM on the target platform, as shown in Figure 6. The technology of JVMs requires that they decode and execute Java programs in real time, and it is constantly modifying the space that the JVM and Java program occupies. This causes problems for a dynamic binary translator because the program is constantly nullifying the translated blocks necessitating repeated re-translation, and therefore not providing the performance demanded of applications.

Transitive has made some technological breakthroughs in the dynamic binary translation of Java programs and their JVMs on the Linux/x86 and Solaris/x86 platforms. Therefore, if there is an incompatibility between the JVM on the

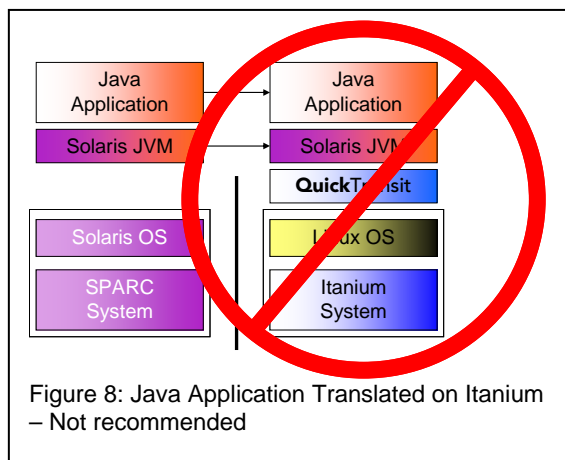


Figure 8: Java Application Translated on Itanium – Not recommended

“Transitive has made some technological breakthroughs in the dynamic binary translation of Java on x86.”

source system and the target system, then reasonable performance can be achieved by running those applications translated with the Solaris/SPARC JVM on the target platform, as shown in Figure 7.

Because of the technology involved in an Itanium platform, translation of Java programs and JVM cannot be made to perform in the same way as the x86 environment. Therefore, as noted in Figure 8, it is specifically **NOT** recommended that you run Java applications and a Solaris/SPARC JVM translated by QuickTransit in a Linux/Itanium environment. The best approach in this situation is to run the Java programs on the native Linux/Itanium JVM, and make whatever changes are needed to have it work in that environment.

Customer Examples

Here are some examples of customers who have migrated their Solaris/SPARC applications to industry-standard platforms using QuickTransit:

- A large semiconductor manufacturer based in the American Southwest migrated Oracle 8 and related applications from a Sun 4/490 server running Solaris 8 to an HP x86 system running Linux. The migration took approximately 48 person-hours of effort to complete.
- A European retailer running SAP® R/3® and Oracle 8 (Solaris 8 running on end of life V-series Sun SPARC servers) was able to redeploy the applications unchanged on Fujitsu-Siemens PRIMERGY® servers running Novell® SUSE® SLES 10 Linux. The move to the new systems took about 3 days. Following a user functionality test the users are happy with the increased performance and the company expects the ROI pay back in less than twelve months.
- A large credit card processing agency in the United States wanted to virtualize a set of Solaris/SPARC systems as well as Windows® and Linux x86 systems onto an x86 system running VMware® ESX. The applications included Apache® Tomcat, Java RMI, Micro Focus™ Cobol, MySQL™ database and UniKix™ Batch Processing Environment. The migration took approximately 64 hours of effort to complete.
- A European IT management and services company providing financial services for large banking enterprises wanted to migrate a set of applications including Oracle 8, JRun™, Apache and FundManager from a set of Sun Blade 100 Solaris/SPARC systems (single core, 500 Mhz) to a single Fujitsu-Siemens system (2 quad-core Intel Xeon® processors) running VMware ESX, Red Hat® Enterprise Linux® x86-64 (RHEL) 4.5 and QuickTransit 1.5.1. The customer reported that the migration went very smoothly and with a minimum of effort.
- Additional customer success examples are available from www.transitive.com or call your local Transitive sales representative (see Next Steps for contact information).

“The customer reported that the migration went very smoothly and with a minimum of effort required.”



Conclusion

The message of this white paper is that there is no migration approach that is “one size fits all” and the approach you use will depend on:

- The system you are migrating from
- The system you are migrating to
- The type of application that you are migrating
- The amount of resources you want to apply to the migration project

The project must approach each application separately and determine the best way to move it to the new environment, or even if it should be moved at all. Then once the details have been determined, look at the overall migration process, timeline, resources required and budget available and determine if it is a viable solution from the perspective of a CIO, CTO or CFO.

Next Steps

To learn more about QuickTransit, visit <http://www.transitive.com>

To evaluate QuickTransit for your own requirements, consult <http://www.transitive.com/evaluate>

QuickTransit is available in a VMware appliance (a pre-configured VM) for convenient evaluation.

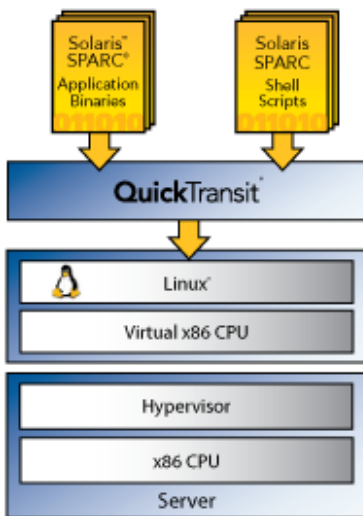
To talk to a QuickTransit sales representative, contact Transitive at sales@transitive.com or you may call our offices at +1 877 399 6111 (US) or +44 (0) 20 7822 4366 (Europe)



TRANSITIVE®

The leader in cross-platform virtualization

Run Solaris™/SPARC® applications in Linux®/x86 or Solaris/x86 VMs with NO changes to source code or binaries



- Breakthrough Performance
- Full Functionality and Transparent Migration
- Improved Application Management
- 100% VMware Compatibility

I|C|E™
systems

ICE Systems Pty Ltd
80 Mount Street
North Sydney, NSW, 2060
Australia
+61 2 9906 1592
www.icesystems.com.au
transitive@icesystems.com.au

Transitive Corporation

Corporate Headquarters
718 University Avenue
Suite 200
Los Gatos, CA
95032-7608

Tel: (408) 399-6611
Fax: (408) 399-6610
Toll-Free: (877) 399-6111

Transitive Limited
Fleet House
8-12 New Bridge Street
London EC4V 6AL

Tel: +44 (0) 20 7822 4366
Fax: +44 (0) 20 7822 4368

www.transitive.com

©2008 Transitive Corporation. All rights reserved. No part of this document may be reproduced, transmitted, or translated without the written permission of Transitive Corporation. Transitive, QuickTransit and the Transitive logo are registered trademarks of Transitive Corporation and/or its affiliates in the United States and other countries. All other company and product names may be trademarks of their respective owners.

TWP093SA-1
November, 2008