Porting Guidelines for Solaris™ Operating System, x86 Platform Edition

An Overview with Real Case Studies
White Paper, April 2004

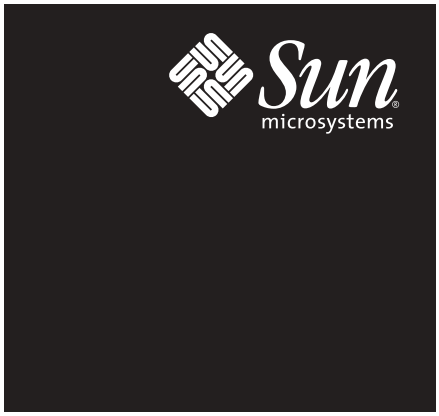

Table of Contents

Chapter 1
Introduction

This document provides an overview of what it entails to port to the Sun Solaris™ Operating System (OS) x86 Platform Edition. This is specifically useful when considering a port from Solaris OS SPARC Platform Edition, or from Linux, to Solaris OS x86. It provides a comprehensive idea about where to look for Solaris OS x86 related modifications. It is not meant to give specifics on modification details.

This document also presents case studies of Independent Software Vendors (ISVs) applications which were ported to Solaris OS x86. It presents the real issues these ISVs faced when porting to Solaris OS x86. It also indicates the effort required and level of ease in porting to Solaris OS x86.

At the end of the document are references that may be helpful in understanding more technical detail about the porting issues.

Chapter 2

The Porting Process

There are areas you need to look into when you are ready to port to Solaris OS x86. This paper can help in understanding the simplicity/complexity of porting to Solaris OS x86. It will also give you an approximate idea of the resource requirements.

When you are ready to port to Solaris OS x86, these are the areas you may want to look into to get further insight into the porting process.

- **Understand the Application Components**
  Are the components in native C, C++, assembly language, and/or Java™ programming language? If all the components are Java™ technology-based, then it is a very EASY port (as good as a "non-port"). Any Java technology-based components will work without any modifications. For example, client GUI components in Java programming language will work as it is on Solaris OS x86 with no changes needed in the source files.

- **Discover the Source Code**
  What are the platform-specific code segments? These segments may need to be modified to support Solaris OS x86. You can use *#define* statements (with "*#ifdef*") to separate Solaris OS x86 platform-specific code segments. You can use *#define* statements for different possible hardware platforms, for example, x86 and SPARC. The same is true for different Operating Systems such as Solaris OS SPARC Platform Edition, Solaris OS x86 Platform Edition, Red Hat, Linux or Windows.

- **Include Header Files**
  If there are specific header files needed for Solaris OS x86, you may need to use *#include* statements in the source files in order to include them.

- **Modify Driver Code**
  If you have a platform-specific driver code, then this may need to be modified to work on Solaris OS x86 Platform Edition. If the Solaris OS SPARC driver is developed using the standard APIs like DDI/DDK, then it takes just a recompile to work on Solaris OS x86. If a driver does not adhere to standard interfaces, then it may need to be modified to make it work on Solaris OS x86. Driver modifications can be minimized by adopting an x86-based Linux driver code for (if this application supports Linux) Solaris OS x86. Due to differences in the driver frameworks of Linux OS and Solaris OS, it may require some modifications in order to port Linux drivers to Solaris OS x86.

- **Modify Hardware-Dependent Code**
  Hardware-dependent code in assembly will need some modifications to work on Solaris OS x86 Platform Edition. For example, assembly code on SPARC hardware will need to be modified to work on the x86 hardware platform. This is due to hardware architectural differences. You can minimize the changes needed to port to Solaris OS x86 by adopting the assembly code from Linux (instead of Solaris OS SPARC Platform Edition).

- **Recompile the Application**
  If porting from Solaris OS SPARC Platform Edition to Solaris OS x86, the application may just need to be recompiled. If the application is developed using published Solaris OS interfaces on SPARC, it does not need any changes.

- **Differences in System/Function Calls**
  If you are porting from Linux to Solaris OS x86, you may need to discover the differences in a few system/function calls used in both these platforms. You will need to discover the equivalent system/function calls used in Solaris OS x86. You can look through Linux man pages for those system/function calls and review the "Conforming To" section. You can also use tools like Autoconf (http://www.gnu.org/software/autoconf/autoconf.html), which can indicate portability issues after scanning through the code, as well as help in creating portable build environments.

- **Changes in Makefile(s)**
  There may be some changes required in the Makefile(s) to generate Solaris OS x86 platform-specific object files, and linking these object files to get the Solaris OS x86 platform binaries. Makefiles may need to be modified to even include appropriate libraries specific to Solaris OS x86. Thus, porting to Solaris OS x86 may affect the build process that you use to generate platform-specific binaries. The tools like Autoconf (http://www.gnu.org/software/autoconf/autoconf.html)or qmake (which generates platform specific Makefiles) (http://doc.trolltech.com/3.3/qmake-manual-1.html) or Jam (http://www.perforce.com/jam/jam.html) can make this process much easier.

- **Third-Party Libraries**
  You also need to discover what third-party libraries (and/or open source libraries) are needed to generate Solaris OS x86-based applications. If these libraries are not available yet on Solaris OS x86, you may want to consider alternate libraries which may already be available on Solaris OS x86.

- **Modify Compiler Flags**
  You may also need to modify the compiler flags used to create optimized executables for Solaris OS x86 Platform Edition. If you are already using Sun Studio or Sun Forte™ tools, then building the application on Solaris OS x86 is much easier. If you are using a different compiler for the Solaris OS x86 platform, you need to consider the appropriate command-line options to use for generating optimized applications for Solaris OS x86.

- **Byte-Order Dependent Code Segments**
  If you are porting your application from Solaris OS SPARC Platform Edition to Solaris OS x86, you may want to look into code segments which are byte-order dependent when stored in the memory. For example, SPARC hardware supports *big-endian* (Most Significant Byte [MSB] is stored in the lowest memory address) whereas x86 hardware supports *little-endian* (Least Significant Byte [LSB] is stored in the lowest memory address).

- **Source Code and Version Control**
  You may also investigate what source code and version control system is currently used. CVS (Concurrent Versions System) and SCCS are available on Solaris OS x86. If the source control system you are using now is not supported on Solaris OS x86, there are various simple workarounds you can use. For example, you can use the source control system on a supported OS, and check out to NFS or Samba (www.samba.org) which will make the files available to Solaris OS x86.

To Summarize:

- Check the source code. If it is Java technology-based then it is a "non-port".
- Verify that third-party (and/or open source) libraries/dependencies are available on Solaris OS x86 platform.
- Modify Makefiles and build process to generate Solaris OS x86-based application. Autoconf, qmake and Jam types of tools can easily generate Solaris x86-based Makefiles.
- If needed, add Solaris OS x86 Platform Edition specific changes using *#define* statements (with "*#ifdef*").
- If needed, add *#include* statements to include Solaris OS x86 specific header files and/or other third-party header files.
- If needed, modify platform-specific drivers for Solaris OS x86 platform. It makes for much easier porting if adopted from Linux-based driver code.
- If needed, modify compiler flags to generate optimized Solaris OS x86 technology-based applications.

Chapter 3

Case Studies: Specific Solaris OS x86 Platform Edition Porting Examples

Leading ISV providing IT solutions in Financial Services Industry

Application:
Application is an industry-leading quote management/market access system that delivers customized, intelligent orders. Its functionality seeks out the best prices and allows users to choose when, where and how quickly they wish to execute trades. It also provides consolidated quotes from various execution venues via direct connections to NYSE, NASDAQ's SuperMontage, the ADF, and all major ECNs for listed and OTC securities.

Ported From/To:
Ported from Solaris OS 8 SPARC to Solaris OS x86 10 build 49 and Solaris OS 9 Update 6.

x86 Hardware:
Sun Fire™ V20z (Opteron 2-way system)

Application's Programming Languages:
C (No assembly).

Total Time to Port to Solaris OS x86:
1 day (not including actual QA time).

Number of Engineers on the Porting Project:
2 engineers.

Changes Needed:
- No changes required in the source files. The port was simply working on the build infrastructure (Makefiles – very few changes) and resolving dependencies. No changes needed in the source files for platform-specific components. The source code was very clean and was ported to multiple platforms. The main business is built on Solaris OS SPARC, but they also have a Linux port for Intel Xeon (which explains why there are no endianness issues) and for Intel Itanium. ISV has written their application from the beginning to be portable.
- Minimized the use of platform-specific code. There are *#ifdefs* in the common code-base to differentiate some function/system calls that are different between the Solaris platform and Linux.
- Modified the build environment to support Solaris OS x86 platform. They have a common Makefile for different platform support. It was a matter of setting variables appropriately. The platform mostly included the build utilities (compiler, linker, etc.).
- They did not use any tools to actually generate the Makefiles. They created a cross-platform Makefile by hand. They used Sun version of the "make" to build.

- Sun™ ONE Studio 8 Compiler Collection was used to build the application. They also were able to build using GNU gcc 3.3.1.
- Sybase Solaris OS x86 header files/libraries, which are now available on Solaris OS x86, were not at the time of this porting. Since the application has dependency on Sybase header files and few libraries, Sybase header files were copied from the cross-platformed Linux.
- During the link phase, there were complaints about the Sybase libraries. The Makefile was changed to do a static link with the Sybase libraries.
- For Sybase setup, installation directory, etc., the same configuration as used on the cross-platformed Linux was used.
- A simple setting of the environment variables in the Makefile was necessary for selecting the appropriate build utilities for Solaris OS x86.

Results:
Very Easy—this was the easiest imaginable port. The code was very portable and well-written. The ISV engineer knew the code, was aware of all dependencies and built the system extremely well.

Content Management ISV

Application:
Content Management Application
It makes it easy to contribute and manage all types of enterprise content, whether it is documents, Web content, images or rich media. This application is integrated with Sun Java System Portal Server.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS x86 9 Update 4.

x86 Hardware:
PC (Dual P4-2400 Mhz, 4 G RAM)

Application's Programming Languages:
Java 2 Platform, Enterprise Edition (J2EE™ Platform) (Java and JSP). No native code.

Total Time to Port to Solaris OS x86:
2 days (0 days to port, 2 days QA).

Number of Engineers on the Porting Project:
2 engineers.

Changes Needed:
- No source code modifications were needed. This involved J2EE code (Java and JSP code). No native code at all. Everything just worked as is on Solaris OS x86 platform. J2EE application was written for portability and tested previously on Solaris OS SPARC.
- No changes needed in the Makefiles or the build environment.
- There was an issue with Installer. This was NOT due to the Solaris OS x86 port itself. This was actually a problem because the port was going from Sun Java System Portal Server 6.1 to Sun Java System Portal Server 6.2 – that is where the installation issues crept in, not from the Solaris OS x86 port.

Results:
Very Easy. (Found no issues, as it was all Java code. It is truly a NON-PORT.)

Metapa – Data Warehousing ISV

Application:
Cluster Database
This is a data warehousing solution used by many large telco customers. This Cluster Database is based on PostgreSQL, which is an open source database.

Ported From/To:
Ported from Red Hat Linux Enterprise Edition 3.0 to Solaris 9 OS x86 Update 5.

x86 Hardware:
Sun Fire V65x

Application's Programming Languages:
C and Assembly.

Total Time to Port to Solaris OS x86:
7 days to port and QA.

Number of Engineers on the Porting Project:
3 engineers.

Changes Needed:
- Used GNU gcc 3.3.2 compiler to port the application on Solaris OS x86.
- Modified the Makefiles to change some compiler flags to optimize on Solaris OS x86. Previously they used Intel compiler for their Linux platform. This switch to GNU gcc compiler required to change compiler flags in the Makefiles. Intel compiler is not yet available on Solaris OS x86.
- They used GNU Autoconf tool to generate Solaris OS x86 platform specific Makefiles.
- No changes were needed in the actual source files.
- No changes were needed in the assembly code segment.
- This application is using an open source product PostgreSQL, which is very portable across multiple OS platforms. The source code is not closely tied to any one architecture and that made it easier to support Solaris OS x86 platform.
- Modified build environment due to a switch to GNU gcc compiler (from Intel compiler).

Results:
Very Easy (just a recompile)

Stellent – Content Management ISV

Application:
Stellent Universal Content Management
It helps in Web content management, document management and collaboration, as well as in records management and digital asset management – with solutions based on one universal technology platform.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS x86.

x86 Hardware:
Sun™ LX50 Server

Application's Programming Languages:
Mostly Java, C++ (Authentication to Apache is in C++.)

Total Time to Port to Solaris OS x86:
Less than a week.

Number of Engineers on the Porting Project:
1 engineer.

Changes Needed:
- Porting to Solaris OS x86 was very smooth, as most of the Java source code needed no changes.
- Added *#include* statements to include certain Solaris OS x86 specific header files.
- They used Ant tool and GNU gcc for compiling.
- Added a new *#define* statement to support Solaris OS x86 platform.
- Most of the time was spent on modifying the build process for the new platform, Solaris OS x86.
- Few changes made in the Makefiles to link appropriate shared libraries related to socket APIs.
- The application has some cryptography-based code, which is byte-order dependent. This part of the source code needed to be modified due to endian-ness issue related to hardware platform differences between SPARC and x86.
- The application seems to be well-designed, i.e. all platform-specific code is grouped together, thus making it easier to add a new platform support using *#define* statement. This made it very easy to port to Solaris OS x86.
- They have same source base for all UNIX platforms (AIX, Solaris OS SPARC, HP-UX, Solaris OS x86, Linux).

Results:
Very Easy (mostly Java code)

Business Intelligence/Data Warehousing ISV

Application:
Business Intelligence Application
Completely Java technology-based. It helps corporate IT departments quickly and cost-effectively deploy business intelligence (BI) solutions to users inside and outside the enterprise.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS 9 x86.

x86 Hardware:
Sun Fire V60x

Application's Programming Languages:
Java programming language.

Total Time to Port to Solaris OS x86:
3 days with QA.

Number of Engineers on the Porting Project:
2 engineers.

Changes Needed:
- This is a completely Java technology-based application, thus needing NO changes in the source files and Makefiles.
- Need to just install the Solaris OS x86 and the application itself.

Results:
Very Easy (It is a completely Java technology-based application. It runs on any platform.)

Power Supply Network Management Application ISV

Application:
Automatic Power Network Control and Management Application
This is an automatic power network control system. The core of the application is a daemon that manages the communication between different applications by socket programming. Database connection modules, power monitor or admin modules use the API provided by the core application to communicate between each other and implement the automatic power network control and management functions.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS x86 9, 04/03 release

x86 Hardware:
Dell PC server

Application's Programming Languages:
C, C++ (Both GUI and network communication components are using C/C++.)

Total Time to Port to Solaris OS x86:
4 days total with QA.

Number of Engineers on the Porting Project:
3 engineers.

Changes Needed:
- They added a new *#define* to support Solaris OS x86 platform.
- Few changes made to the source code, which is byte-order dependent. Some small changes such as the byte order of a variable stored in the memory between x86 and SPARC platform are reversed (endian-ness issue). They solved this issue by using *#define* to distinguish the differences among different platforms.
- The application was using GNU gcc 2.95.3 and Qt (http://www.trolltech.com/products/index.html) as their development platform. Qt generates platform-specific Makefiles. Thus, Makefiles needed no changes as Qt (qmake) implements the platform-specific changes automatically. For more information on qmake, please refer to http://doc.trolltech.com/3.3/qmake-manual-1.html
- They changed certain compiler flags when porting from Solaris OS SPARC to Solaris OS x86 to optimize on Solaris OS x86 platform.
- Modified build process to support Solaris OS x86 platform.

Results:
Easy (The porting is easy because their application is using GNU gcc and Qt solution. They use Qt from http://www.trolltech.com/products/solutions as their development framework, so it is easy for them to port to a new Solaris OS x86 platform.)

Financial Management Application ISV

Application:
Financial Management Application
It is a centralized financial management product to strengthen management accounting functions and provide various solutions for different industries.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS x86 9, 04/03 release.

x86 Hardware:
LX50(1xCPU), Dell PC

Application's Programming Languages:
Java programming language (all components).

Total Time to Port to Solaris OS x86:
6 days (mostly due to QA process)

Number of Engineers on the Porting Project:
2 engineers.

Changes Needed:
- No changes needed in the source files, not even for platform-specific changes. This is because the application is a Java application and can run on any platform.
- No changes needed in the Makefiles.
- There were some minor configuration changes and environment setup changes to support Solaris OS x86 platform.

Results:
Very Easy (because the ISV application is completely written in Java programming language. This is truly a "NON-PORT".)

DeepNines - Intrusion Prevention Application ISV

Application:
DeepNines Sleuth9 Intrusion Prevention Application
DeepNines integrates system vulnerability analysis, intelligent firewall, intrusion prevention, best-in-class anti-virus functionality, spam filtering as well as forensic capture and reporting into a security software platform. Sleuth9, the company's patent-pending security system, is a fully automated intrusion prevention system. Sleuth9 can be deployed in front of the router, thus preventing malicious code from damaging enterprise networks and improving organizations' security deep into the nines.

Ported From/To:
Ported from Solaris OS SPARC to Solaris OS x86 9 U4.

x86 Hardware:
Sun Fire V60x and Sun Fire V65x servers

Application's Programming Languages:
C (All source code is in C.)

Total Time to Port to Solaris OS x86:
2 months to port and QA.

Number of Engineers on the Porting Project:
3 engineers.

Changes Needed:
- Its main component is the kernel module which was recompiled on Solaris OS x86. It is mainly a recompile from Solaris OS SPARC to the Solaris OS x86 platform.
- Some changes were made to the source code where byte order was important. This is to take care of endianness issue due to hardware architectural differences between SPARC and x86 platform.
- Sun ONE Studio 7 compiler was used for SPARC and the same was used for Solaris OS x86 platform. This makes it easier to recompile as the same tool takes care of platform-specific flags.
- Changed some compiler flags to optimize for Solaris OS x86.
- Added *#define* to support Solaris OS x86 platform.
- This application has a dependency on an anti-virus engine (database/lib/API) which is from a third-party vendor and is not yet available on Solaris OS x86. Workaround is to find another Solaris OS x86 platform-based anti-virus engine.

Results:
Easy (mostly a recompile)

Sybase – Information Management for Enterprise and Mobility – ISV

Application:
Adaptive Server Enterprise ASE 12.5.1
(http://www.sybase.com/products/databaseservers/ase) Sybase Adaptive Enterprise is a powerful data management platform for high-performance business applications.

Ported From/To:
Ported from Solaris OS SPARC to Solaris 9 OS x86 Update 4. Any architectural-specific assembly code was taken from their Linux product.

x86 Hardware:
Sun Fire V60x, Dell 6650

Application's Programming Languages:
ASE engine: C, Java, Assembly code. Tools were developed using Motif libraries in C and Java programming language.

Description of the Functionalities:
ASE is comprised of two major components: Open Client Server (OCS) and the ASE kernel. The Open Client libraries are required by ASE to function. OCS is divided into open server (gateway to connect to ASE engine) and open client (ability for client to communicate to ASE). These are primarily developed using C. ASE kernel is the actual database engine and is developed with C and assembly.

Total Time to Port to Solaris OS x86:
6 months total for port and QA (Sybase has a very rigorous and methodical QA and Product Release process for their products and that is the reason for this rather long time frame).

Number of Engineers on the Porting Project:
4 engineers.

Changes Needed:
It was a straightforward re-compile from Solaris OS SPARC to Solaris OS x86:

- Java components were not modified.
- Fibre channel drivers were not available on the Solaris OS x86 platform at the time when they started on this porting project, thus a SCSI driver was used as a workaround.
- Sun™ Cluster technology, which is now available on Solaris OS x86, was not available at the time of the porting, so support was not needed.
- There were some changes needed due to hardware differences between SPARC and x86, especially related to endian-ness and ASE error handling that was tight to hardware.
- Added *#define* statement (with "*#ifdef*") to make Solaris OS x86 platform-specific changes.
- There was some source code in assembly language. This was modified for the Solaris OS x86 platform. It was difficult to move assembly code from Solaris OS SPARC to Solaris OS x86, so it was decided to port assembly code from their Linux-base to Solaris OS x86 platform.

- Sun Forte 6 Update 2 C/C++ Enterprise Edition (x86) compiler was used to compile the code on Solaris OS SPARC, as well as on Solaris OS x86. Few compiler flags were modified to optimize the application for the Solaris OS x86 platform.

Results:
Porting was relatively easy – mainly involved recompiling and code changes that are hardware-related from the porting perspective. Difficulties were mainly due to non-availability of some drivers and features at the time of the porting process.

Large ISV in Financial Services Industry

Application:
Fixed Income Application
The application models the value of bonds and this application is used by many large financial services companies.

Ported From/To:
This application is ported from Solaris OS SPARC 2.6 to Solaris 9 OS x86.

x86 Hardware:
Sun Fire V20z Server (Opteron 2-way system)

Application's Programming Languages:
C (for mathematical modeling and networking functionalities).

Functionalities:
It uses caching of models for better performance. It goes out to its database server to get mathematical models. Often the same models can be reused, so it caches them in binary data structures locally to its application server to save the database fetches.

Total Time to Port to Solaris OS x86:
1 week.

Number of Engineers on the Porting Project:
3 engineers.

Changes Needed:
- *#define* statements added to support new Solaris OS x86 platform.
- *#include* statements added to include Solaris OS x86 specific header files especially from a third-party vendor.
- Needed to modify few source files related to byte swapping functionalities due to differences in x86 hardware platform compared to SPARC hardware (to take care of endian-ness issue). For Solaris OS x86 platform they have to byte swap the data because they "mmap"-ped some binary files that had been written from a SPARC architecture.
- Added some runtime detection for the caching code. It tests some code at run time to see if it had to be byte swapped for endian-ness rather than doing so at compile time with "#ifdef" or similar pre-processor code.
- Original code was written with old Sun Studio compilers from the Solaris OS 2.6 era (pre Forte). Code was then ported to gcc 3.3. Sun Studio could not be used due to word alignment differences in x86 and SPARC hardware. SPARC compilers align structure fields to a double word boundary, while Solaris OS x86 Sun Java Studio compilers align them to a word boundary. Due to this limitation of Sun Studio, gcc has to be used which can be told to compile on a double word boundary with: "-malign-double".
- Port was done using GNU gcc 3.3 compiler, which required few Makefile changes, especially related to compiler flags. Some flag changes were needed to produce position-independent code for the shared libraries.
- Some of the compiler/linking issues were induced by the compiler change (to gcc). Using gcc, the structures had to be prefixed by & (address of) to pass by reference.
- Gcc also required *#define* "NULL 0". Without this *#define*, "NULL" was generating some type checking errors in gcc.

- Array initialization was not the same across compilers and required some changes.
- This application had a Sybase dependency needing Solaris OS x86 Sybase client libraries. These Solaris OS x86 libraries are available from Sybase.
- Only major issues were Sybase dependencies outlined above and word alignment issues. Since the application shared data between SPARC database servers and x86 application servers, which was written in binary form to disk as a cache, alignment issues within the application structures were critical. This forced a port using gcc (which was being considered for performance reasons because of SSE support) as well as gcc supports "-malign-double"
- Source code control system used was Rational Clearcase. The ISV was in the process of making a change to CVS (from Clearcase) to save some cost. In addition, CVS is available on Solaris OS x86 platform.

Results:
Medium difficulty in porting

Chapter 4
Conclusion

This gives you an overview of what it entails to port applications to the Solaris Operating System x86 Platform Edition. It focuses on areas you may want to investigate before you actually do the porting. This paper was designed to help developers and managers in their product planning process, especially about resource needs and the scoping of the porting project.

Looking at the case studies, we can conclude that in most of the cases, porting to Solaris OS x86 is a very easy process. This is especially true if you have an application that is written completely in Java programming language. In this case, it is as good as a "non-port" because it is truly a "write once, run anywhere" application.

You can expect to spend more time tweaking your Makefile(s) to include Solaris OS x86 related header files and libraries, as well as modifying certain compiler flags to get the best performance on the Solaris OS x86 platform. Similarly, you will spend some time modifying your existing build process to generate executables for the Solaris OS x86 platform.

Most importantly, ascertain if the application has any dependencies. Find out the availability of these dependencies on Solaris OS x86. If these dependencies are not yet available on Solaris OS x86, you may want to consider some workarounds.

Chapter 5
References

Guide to Adding Support for Solaris OS, x86
http://developers.sun.com/solaris/articles/support_for_x86.html

Compiler Differences
http://developers.sun.com/solaris/articles/x86_compiler_diffs.html

Case Study: Porting Apache Web Server from Solaris OS SPARC to Solaris OS x86
http://developers.sun.com/solaris/articles/casestudy_port_x86.html

Migration to Sun Platforms for Developers
http://iforce.sun.com/partners/migration/

Solaris Operating System for x86 Platforms
http://wwws.sun.com/software/solaris/x86/index.html

Solaris OS x86
http://www.solaris-x86.org

Sun Software Express Program
http://wwws.sun.com/software/solaris/solaris-express/index.html

Hardware Compatibility Lists
http://www.sun.com/bigadmin/hcl/

Tunathon
http://sdc.sun.com/tunathon2004/

Solaris OS Forums
http://forum.sun.com

System Administration Portal
http://www.sun.com/bigadmin/

Community SW Packages for Solaris OS
http://www.blastwave.org/

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com

Sun Worldwide Sales Offices: Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323 Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0 Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India–Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500 Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100 The Netherlands +00-31-33-45-15-000, New Zealand–Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China–Beijing +86-10-6803-5588; Chengdu +86-28-619-9333 Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Saudi Arabia +9661 273 4567, Singapore +65-6438-1888 Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland–German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888 Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800, or online at sun.com/store

Printed in USA SunWIN #409702 05/04