Preparing HP-UX Software for IA-64

Debbie Lienhart
Hewlett-Packard Company
3404 E. Harmony Road, MS 7
Fort Collins, CO 80528-9599
(970) 898-4227
(970) 898-7734 (FAX)
debbie@fc.hp.com

IA-64 is Intel's next generation 64-bit architecture for microprocessors. It contains the 64-bit Instruction Set Architecture (ISA) developed jointly by Hewlett-Packard Company and Intel Corporation, as well as an IA-32 compatible component. (IA-32 is Intel's 32-bit architecture, also known as x86. IA-32 chips include the 486, Pentium®, and Pentium II®.) The 64-bit ISA is based on the EPIC architecture technology, also jointly defined by HP and Intel. EPIC stands for Explicitly Parallel Instruction Computing. IA-64 supports both 32-bit and 64-bit computing environments and is compatible with HP's existing PA-RISC (Precision Architecture Reduced Instruction Set Computing) architecture.

HP-UX on IA-64

HP-UX for IA-64 is a version of HP-UX 11.x. The same HP-UX version will run on both PA-RISC and IA-64 systems, and will be compiled from common source for compatibility. The environment (commands, utilities, user interfaces, shell scripts, and so on) will be the same on PA-RISC and IA-64 systems. IA-64 supports both big Endian and little Endian operating systems. HP-UX has always been a big Endian operating system, and will continue to be so on IA-64. This means that HP-UX applications will be able to read their existing binary data on IA-64.

There are a few planned differences between HP-UX on PA-RISC and IA-64. In particular, HP-UX on IA-64 will have a different run-time architecture, object file format, and debug information format. The different run-time architecture is needed to support IA-64. The new object file and debug information formats will make HP-UX more similar to other UNIX operating systems. These new formats will make it easier for software development tool providers to support HP-UX, and should result in more software development tools for HP-UX. The HP-UX implementation will also have some minor changes to support IA-64. For example, the kernel data structures will need to be updated to handle all of the registers on IA-64. These changes should only affect kernel-intrusive software.

Software Compatibility

Well-behaved PA-RISC binaries that run on HP-UX 11.x will run on HP-UX on IA-64. The new IA-64 architecture was designed to enable binary compatibility with the PA-RISC architecture. Source compatibility will also be provided between HP-UX 11.x on IA-64 and HP-UX 11.x on PA-RISC.

In general, well-behaved applications are those that are not sensitive to the architecture or operating system implementation. Well-behaved applications follow these general guidelines for compatibility:

- Use only public APIs.
- Adhere to required practices that are specifically documented.
- Do not use features that are specifically described as having platform, architecture, or configuration limitations.
- Do not decompose an HP-UX product and then re-use the results of the decomposition.

These guidelines are described in more detail in the Coding Practices for Compatibility paper, which can be found in the HP-UX 11.x Software Transition Kit.

Deciding Whether to Run PA-RISC or Native IA-64 Binaries

Both well-behaved PA-RISC and native IA-64 HP-UX binaries will run on HP-UX on IA-64. Native IA-64 binaries will run faster because they will be compiled to expose the parallelism in the code and will be optimized to take advantage of the architectural features and resources of IA-64.

Many applications will have acceptable performance without being recompiled for IA-64. The capability to run both existing PA-RISC and native IA-64 binaries allows HP customers and ISVs to make an incremental transition to HP-UX on IA-64, as illustrated in the following two examples.

In the first example, suppose you want to run several of your applications on an IA-64 based system. You can start by using native versions of just the architecture-sensitive applications, while running all other well-behaved applications in compatibility mode on IA-64. You can then update the latter applications to native versions over time. The primary benefit of this approach is that you need not wait until you have native versions of all your applications to make them available on IA-64. By changing one application at a time, it will also be easier to diagnose any problems that may occur during the transition. Plus, system performance will improve as more of your applications become native.

For the second example, suppose you have an application that is composed of many processes, and the different processes have varying levels of architecture- and performance-sensitivity. In this case, you can start by compiling native versions of just the architecture-sensitive processes of your application, while leaving the rest to run in compatibility mode. This allows you to provide the application to your customers as early as possible. You can then compile native versions of the performance-sensitive processes in order to provide a high-performance version to your customers. Eventually, you might want to compile native versions of all the processes, if an all-native version of your application would be easier for you to build and support long-term.

As you can see from these examples, you have a great deal of flexibility when transitioning your software to HP-UX on IA-64. You can either leave your well-behaved PA-RISC binaries running in compatibility mode on IA-64, or re-compile them into native IA-64 binaries. You also have the option of transitioning processes incrementally within an application to native IA-64. This inherent flexibility allows you to get up and running on IA-64 much faster on HP-UX than with other enterprise UNIX operating systems.

HP-UX 11.x Software Transition Kit

The HP-UX 11.x Software Transition Kit (STK) is the foundation for HP's IA-64 software transition tools and processes. You can use the STK now to prepare most of your existing PA-RISC binaries and source code for HP-UX on IA-64. HP continues to add information about transitioning software to IA-64 to the STK as it becomes available.

As explained earlier, well-behaved PA-RISC binaries that run on HP-UX 11.x will also run on HP-UX on IA-64. To make your well-behaved PA-RISC binaries ready for IA-64, you can test them on HP-UX 11.x on PA-RISC with the qualification process described in the STK.

How to Prepare Your Source Code for HP-UX on IA-64

Native binaries will provide the best performance on the IA-64 architecture. You can create a native binary by compiling on IA-64. The version of HP-UX fully supporting IA-64 will be source-compatible with the current HP-UX version 11.x. Thus, the following subsections outline the steps you can take now on HP-UX 11.x to make your source code ready for IA-64. Exceptions are listed in the section Software Which Cannot be Pre-Enabled on HP-UX 11.x.

Resolve IA-64 Transition Issues

The first, key step in preparing your source code for IA-64 is following the porting processes (investigating, planning a port, and performing a port to HP-UX 11.x) that are documented in the HP-UX 11.x Software Transition Kit (STK). You can use the **`scansummary`** and **`scandetail`** file scanners included in the STK to identify transition issues, such as API changes, in your source code files.

To identify the planned differences between PA-RISC and IA-64 that will affect your source code or binaries, you will need to scan your source code files using the **IA64** classification option with the file scanners. For example, to get a summary output report listing only the IA-64 API transition issues in a particular source code file, run the **scansummary** tool on the command line like this:

```
scansummary +C IA64 source_file
```

Once you have run the file scanners to identify all the IA-64 API transition issues in your source code files and have completed a porting plan, follow the porting process to resolve the necessary transition issues in your source code. Pay special attention to resolving any Non-critical Non-standard (**NcNs**) impacts in your source code for APIs that are likely to cause compatibility problems.

If you cannot resolve an API impact that is specific to IA-64, you may not be able to fully prepare your source code for IA-64 at this time. In this case, resolve all the transition issues you are able to now, keeping track of the transition issue(s) that will require more work later.

Use Kernel Threads Instead of DCE Threads

HP-UX DCE threads were based on an early draft of POSIX threads, and there are some significant differences between that draft and the final standard, which is implemented as the kernel threads set of APIs. Like DCE threads, kernel threads allow you to develop multi-threaded applications. However, kernel threads provide finer granularity for optimizing performance in applications that can exploit a threaded architecture. HP-UX 11.x contains thread-safe libraries for C++, Pascal, math/vector, networking, and the dynamic loader, based on the POSIX 1003.1c standard.

DCE threads will still be available for IA-64, though they may be obsoleted in the future. For more information on using kernel threads, see the paper Introduction to Kernel Threads in the STK.

Use Shared Libraries

Shared libraries are preferred over archive libraries during the transition to IA-64 for the following reasons:

- Shared libraries avoid binding architecture dependencies into the application. This is especially important for PA-RISC applications running on IA-64.
- Defect fixes are picked up automatically when a library is patched.
- The performance difference between archive and shared libraries is decreasing as new run-time architectures are optimized for shared libraries.

Use `pstat(2)` to Read System Files

If any of your code reads system files directly, such as **/dev/kmem**, revise your code to use **pstat(2)** routines to get system information instead. This is because system files will likely have different formats on IA-64 based systems. By using **pstat(2)**, information can be retrieved about the following:

- Memory -- static, dynamic, shared, virtual
- System CPU processors
- System processes and messaging
- Disks, swap space, and file systems
- Semaphores

See the **pstat(2)** man page for more information. Note**:** The **pstat(2)** routines are HP-specific routines and are not available on other vendor platforms.

Also, if you have 32-bit applications that you will deploy on IA-64, you must use the **pstat()** wrappers turned on with **-D_PSTAT64** to function correctly.

Make 64-bit Applications 64-bit Clean

Many operating systems that currently have only 32-bit support, including Microsoft Windows®, will be available in 64-bit versions for the first time for IA-64. While HP-UX and these other operating systems

will continue to provide support for 32-bit applications, you may want to take this opportunity to create a 64-bit version of your application. Since HP-UX 11.x already provides 64-bit support, you can make your software ready for 64-bit mode on HP-UX now so you do not encounter as many changes later.

Different data models are used in 32- and 64-bit UNIX computing, including HP-UX. The data model used in 32-bit UNIX computing is called ILP32, meaning that integers, longs, and pointers are all 32-bit data types. The data model used in 64-bit UNIX computing is called LP64, meaning that longs and pointers are 64-bit data types, while integers remain as 32-bit data types. Some existing C and C++ code assumes that integers, longs, and pointers are the same size. Because these assumptions are not true in LP64, some executables will have defects when they are compiled for 64-bit mode.

Problems can arise, however, when you try to make 32- and 64-bit applications interoperable. The difference in the size of the ILP32 and LP64 data models and in system address space can create problems when data is transferred from a 32-bit application to a 64-bit application or operating system, and vice versa. These differences can adversely affect the sharing of data between applications, sharing of data between an application and the operating system, communication between processes, working with large files, loading processes, and working with graphical interfaces. For more information, see the white paper Interoperability of 32- and 64-Bit Applications on 64-Bit HP-UX in the STK.

Tools and Guidelines for Making Source Code 64-bit Clean
If any of your applications will be compiled for 64-bits, they should be 64-bit clean. Tools to help make your code 64-bit clean include using **`lint`**, compiling in ANSI C or aC++, and using the portable header file **`<inttypes.h>`**.

The following guidelines will also help make your code 64-bit clean:

- Use the same source code and header files for both 32- and 64-bit applications.
- Use appropriate data types consistently and strictly. For example, use **`off_t`** consistently for file offsets and **`fpos_t`** for file positions.
- Use integral types in **`<inttypes.h>`**, where applicable, instead of **`int`**s and **`long`**s.
- Use fixed/scalable width integral types, algorithms, and masks as appropriate. Fixed types remain a consistent size on 32- and 64-bit platforms. For example, use **`int32_t`**, defined in **`<inttypes.h>`**, if **`int`**s and **`long`**s should be 32-bits in your application. Scalable types can grow and scale to future architectures, such as IA-64, without source code modifications.
- Perform boundary checking on integral type ranges.
- Update 64-bit code in cases where 32- and 64-bit processes share the same memory segment.

Compile on HP-UX 11.x

Finally, you can make most of your source code ready for IA-64 by compiling it on HP-UX 11.x. The IA-64 compilers will be compatible with the current HP-UX 11.x compilers. HP will provide the C, ANSI C++, COBOL, Fortran90, and Java compilers for HP-UX on IA-64. The C, ANSI C++, and Fortran90 compilers will generate both 32- and 64-bit code. The Cfront C++, Fortran77, and Pascal compilers will be obsoleted for IA-64, so you should move code off them now. Note, however, that the Cfront C++ run-time environment will still be available for running any existing (PA-RISC) Cfront binaries you may have.

The C compiler and C Lint in HP-UX 11.x have a +M1 option that you can use to identify compiler and linker options that will either change or be obsoleted for IA-64.

Optimization will be increasingly important on IA-64. The IA-64 compilers will use optimization techniques based on the current PA-RISC compilers. Therefore, you should experiment with higher levels of optimization and Profile Based Optimization (PBO). Optimizing now will not only expose defects in your code, but it will also result in a faster PA-RISC application.

Software Which Cannot be Pre-enabled on PA-RISC

Most source code can be made ready for IA-64 now on HP-UX 11.x on PA-RISC. However, software with either assembly language code or kernel-dependent code, or any software that is sensitive to either the different run-time architecture, the new 32-bit object file format, or the new debug format of IA-64 cannot be pre-enabled now.

This section describes these exceptions. If you have source code that you cannot pre-enable for IA-64 now on HP-UX 11.x, contact HP via the HP Developer's Resource web site (http://devresource.hp.com).

Assembly Language Code

You cannot pre-enable applications with assembly language code because the new IA-64 instruction set is vastly different from the PA-RISC instruction set. If you have any assembly language applications needing to be compiled on native IA-64, you must re-code and performance tune the assembly language source.

Kernel-Dependent Code

The HP-UX kernel will require changes to support IA-64 fully. This may cause changes to kernel drivers and other code accessing kernel data structures. Hence, if your code is kernel-dependent (for example, kernel drivers), you cannot pre-enable it on HP-UX 11.x.

Software Sensitive to the Different Run-time Architecture

The IA-64 run-time architecture is different from that on PA-RISC. If you have software that is sensitive to the run-time architecture, you will need to update it for IA-64 and then compile it on native IA-64.

Software Sensitive to the New 32-bit Object File Format

The HP-UX object file format for IA-64 will be based on ELF. ELF (Executable Linking Format) is an industry-standard object file format.

The current 64-bit version of HP-UX 11.x on PA-RISC uses the ELF-64 object file format. ELF-64 is, for the most part, a simple extension of the ELF-32 format, as originally defined by AT&T Corporation.

For 32-bit object files on IA-64, HP-UX will change from SOM object file format to ELF-32 object file format. The SOM (System Object Model) format is a 32-bit, HP-proprietary object file format for all releases of HP-UX on PA-RISC. Although moving to the ELF-32 object file format will require changes to software that manipulates object files, it will be easier for you to maintain this software on multiple UNIX systems in the future.

Software Sensitive to the New Debug Format

Finally, the format for debug information will also change for IA-64. The new format should be easier to work with, but the change will affect debuggers and other software that read and write debug information. Therefore, such software cannot be pre-enabled for IA-64 now on HP-UX 11.x.

Conclusion

HP-UX systems based on Itanium, Intel's first IA-64 processor, will soon be available. These systems will run both native IA-64 HP-UX binaries, as well as most PA-RISC binaries. In either case, you can use the HP-UX 11.x Software Transition Kit (http://devresource.hp.com/STK) to prepare your software now.