X-station X11perf Results

Another benchmark that is used to compare X11 performance on X-stations and workstations is X11perf. The X11perf benchmark is described in Chapter 5 and the operations covered by X11Perf are listed in Appendix D.

These results are based on the Beta B.04 release of the software and are subject to change with the final release of software.

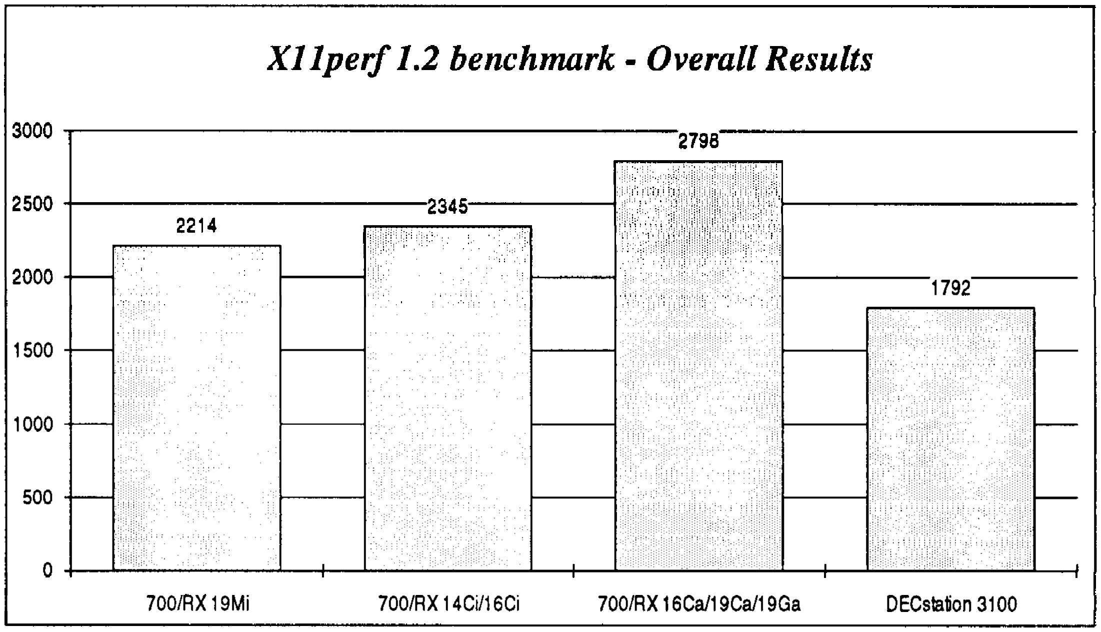

Figure 8.2

X11Perf Composite Results

	19Mi	14Ci, 16Ci	16Ca, 19Ca 19Ga	DECstation 3100
General Graphics	2193	2500	2989	1537
Terminal Emulation	15533	11577	19601	11498
Window Management	1466	869	977	852
X Specific Operations	1900	2291	2513	2512

Table 8.1

Note: Digital Review reports X-station numbers relative to the DECstation 3100 workstation. Results quoted in that magazine are referred to as DXUP's. This is the ratio of the overall geometric mean of the X-station to the DEC 3100. DXUP values can be calculated from Figure 8.2 and Table 8.1.

The B.04 Release of the X Server Software

New Features

There are several new features included in the B.04 software release that will have an effect on the host system as well as the HP 700/RX-station. The features include the ability to run a local window manager, local terminal emulators and a local user environment.

Use of these features will require additional X-station memory, but will reduce the host system memory and system resource requirements. The net effect is that a given host computer system can support more X-stations. The number of X-stations per host system is highly application-dependent.

X-Station Memory Utilization

The following tables will help to estimate the required X station memory.

Local Window Manager [1]	Kbytes Needed
TWM [2]	392
MWM [2]	1656
VUE/RX [3]	2433
VUE WM [4]	2873

Local Clients [1]	Kbytes Needed
Hpterm first invocation	1126
Hpterm second invocation	324
Xterm first invocation	644
Xterm second invocation	185

X Server [1]	Kbytes Needed
Monochrome 19Mi	2400
Mid Range 16Ci	2600
High Performance 19Ca, 19Ga	2800

1. These results are based on a Beta B.04 release of the software and are subject to change with the final release of software.
2. Alternate window managers; not used on the X-station if either VUE option is selected.
3. New GUI environment with multiple workspace manager that is similar to HPVue but is not host dependent.
4. The HPVue window manager only runs on the HP 700/RX-station. The other parts of HPVue such as File Manager runs on the host system.

For example, to run VUE/RX plus 3 HPTERM windows the X-station would require a minimum amount of memory equal to 2433 (VUE/RX) + 1126 (HPTERM 1st) + 648 (HPTERM 2nd x 2) = 4207 Kbytes. To this number, the X server code and memory requirements for the applications must be added. Therefore, it is recommended that a total of 10 Mbytes of X-station memory be installed in the HP 700/RX when running local clients.

Chapter 9 - Advanced Optimizing Compilers for PA-RISC

The Optimization Performance Edge

Compiler optimization technology unlocks the performance potential of the PA-RISC architecture. The RISC philosophy has fundamentally changed the role of the compiler. As RISC moves to strike a balance between hardware and software that exploits the best of each technology, the resulting simple, high-performance instruction set gives the compiler more opportunity to apply optimizations that dramatically improve performance. In fact, this opportunity is more of a responsibility. The effectiveness of RISC depends on the compiler's ability to create the most efficient instruction sequence by appropriately rearranging the program steps. Without these optimizations, many applications will execute at a performance level far below their potential.

Indicative of this synergy, PA-RISC depends heavily on the compiler's optimization technology to maximize performance. The PA-RISC compilers offer the level of optimization sophistication required to deliver industry-leading RISC performance. As illustrated in Figure 9.2, the significant performance boost over unoptimized code reflects optimization techniques designed to extract the performance and efficiency inherent in the PA-RISC architecture. For example, on the Model 720 the HP-UX 9.0 optimizations increase the SPECmark from 21.6 to 66.5 (207%), the MIPS rating from 36 to 57 (58%), and the MFLOPS rating from 2.9 to 17.9 (517%).

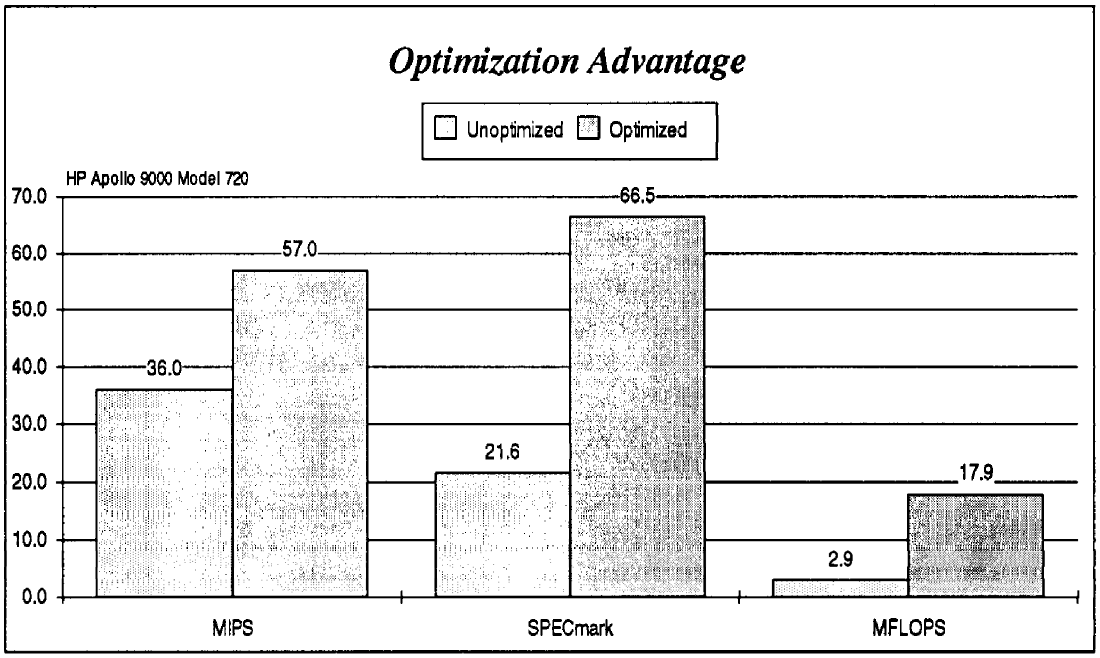

Figure 9.1

Advanced Optimization Technology

The PA-RISC compilers implement a comprehensive set of optimizations geared towards delivering superior performance on a large class of applications running on the PA-RISC processor. Recent optimizer enhancements include the following:

1. Improved Instruction Scheduling: The scheduling heuristics were enhanced to enable the scheduler to generate more optimal schedules, especially systems based on the new superscalar PA7100 processor.
2. Improved Register Allocation: The register allocation algorithm was enhanced to minimize the number of spill instructions generated when the allocator runs out of machine registers.
3. Improved Software Pipelining: Numerically intensive applications often have loops that contain long-latency operations, such as memory accesses and floating-point operations. The compilers use a scheduling technique called software pipelining that overlaps operations from multiple loop iterations in order to hide the long latencies. Enhancements for HP-UX 9.0 include greater integration with register reassociation.
4. Improved Register Reassociation: This optimization reassociates calculations in array subscript expressions to produce more candidates for code motion and strength reduction. Enhancements include increasing the scope beyond just simple innermost loops, as well as greater integration with software pipelining.
5. Improved Branch Optimization: Additional branch optimizations were implemented to minimize the branch penalty along the most frequent path taken. Based upon information from heuristics used to estimate which branches are taken most often, code is repositioned so as to eliminate some of these branches.
6. Profile Based Optimizations: Using an application's execution-profile data, the optimizer rearranges code to improve the instruction cache locality as well as to reduce the number of branches taken. Improving the locality of frequently invoked subroutines results in better utilization of high-speed cache memory.

The continuous enhancement of HP's compilers, in concert with hardware and system software advancements, results in increased performance for each release.

For example, Figure 9.2 shows performance on a Model 720 for successive releases of HP-UX. The SPECmark benchmark performance increased approximately by 36% from HP-UX release 8.01 to 8.05. Performance improved by 12% from HP-UX release 8.07 to 9.0 due to optimizer enhancements.

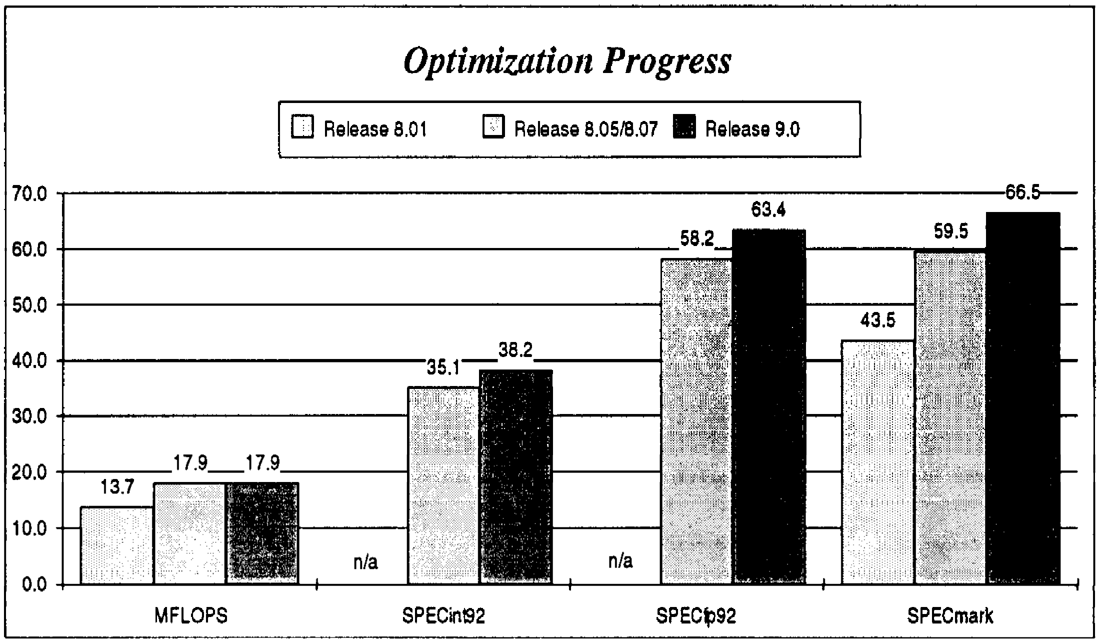

Figure 9.2

FORTRAN Optimizing Preprocessor

The FORTRAN Optimizing Preprocessor analyzes the program by gathering information about the use of data and the nature of the control flow in the program. It then uses this information in conjunction with parameters that describe the machine (such as cache capacity, cache line size, number of registers) in order to restructure source code to achieve data locality.

For example, one restructuring technique called "blocking" treats array operations as multiple operations on blocks of the array, where the size of the block is chosen carefully to match the machine characteristics. This technique can significantly reduce the number of memory loads and cache misses. A number of commonly used numerical algorithms, such as matrix multiplication, benefit substantially from this technique.

Activating Optimizations

The default for PA-RISC compilers is to not optimize your program. Optimizations are activated by specifying particular command-line options. As the analysis required to optimize programs requires additional time and space, the PA-RISC compilers support various levels of optimization for you to control trade-offs between compile-time overhead and code performance improvements.

Table 9.2 provides a summary of optimization options for the C and FORTRAN compilers. For further details, see the Reference Manual and Programming Guide for the five programming languages supported on PA-RISC: C, C++, COBOL, FORTRAN and Pascal.

Table 9.2
C and FORTRAN
Optimization Options Summary

Option	Description
+O1	Instruction scheduling and a subset of optimizations performed on small subsections of code.
-O or +O2	Includes '+O1', plus optimizations performed over the entire scope of each subroutine. This is the generally recommended level for all modules.
+O3	Includes '+O2', plus linker optimizations to improve global variable accesses.
+OP	Invokes the FORTRAN Optimizing Preprocessor (as described in FORTRAN Optimizing Preprocessor).
+I ,+P	Enables profile-based optimizations (as described in Advanced Optimization Technology).

Compiling and Optimizing for Different PA-RISC Architectures

As the PA7000 architecture of the Series 700 workstations differs slightly from the PA-RISC 1.0 architecture of many of the HP 9000 Series 800 systems, code generation and instruction scheduling can be optimized for each architecture. The compilers support the following compile-time options to control these algorithms:

+DS1.0	Instruction scheduling optimized for PA-RISC 1.0.
+DS1.1	Instruction scheduling optimized for PA-RISC 1.1.
+DA1.0	Instruction set of PA-RISC 1.0.
+DA1.1	Instruction set of PA-RISC 1.1.

The default code generated for Series 700 is '+DS1.1 +DA1.1'; and the default code generated for Series 600/800 is '+DS1.0 +DA1.0'. While code generated with any combination of these options will run on the Series 700, the reverse is not true. Code for Series 800 systems based on PA-RISC 1.0 (that is, all models, except the recently introduced 8*7 models) must be compiled using the PA-RISC 1.0 instruction set ('+DA1.0').

A system model number can be used instead of the implementation number (that is, '+DA855' instead of '+DA1.0'). If the program is to be run on one particular target system that is a different HP9000 model than the system used for compiling, then +DA and +DS should be used with the target system's model number. If the program is to be run on many models of the HP9000 (including Series 800 systems) then '+DA1.0' can be used to ensure portability and +DS used with the model number of the fastest hardware system.