Performance Aspects of Portability

By Steven W. White and John R. Reysa

The recently announced POWER2 and PowerPC systems extend the RISC System/6000 family of workstations and servers, which began with the introduction of POWER-based systems in 1990. This article contrasts the architectural and implementation differences for the POWER, POWER2, and PowerPC models while focusing on the performance aspects of compatibility. It also examines the performance gains associated with several compiler optimizations and describes a strategy for selecting compiler tuning options.

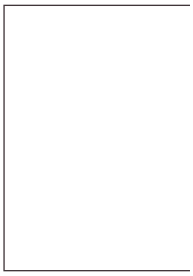
POWER, POWER2, and PowerPC Designs

The classic POWER-based systems have improved dramatically in the three years since 1990: clock rates have more than doubled, instruction caches have quadrupled, I/O subsystem bandwidths have increased, and compilers exploit the hardware to a greater degree. These changes have tripled performance on LINPACK, a frequently quoted benchmark, from 13.0 MFLOPs to 38.1 MFLOPs during the three years. Each of the POWER-based systems include a single fixed-point pipeline, a single floating-point pipeline, and a single branch unit.

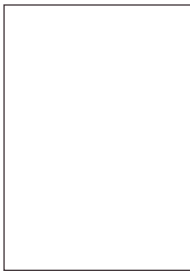
Compared to POWER-based systems, the POWER2 Models 58H, 590, and 990 expand the data cache capacity to 256 KB, double the number of functional units, double the bandwidth of most buses, and quadruple the cache to Floating-Point Unit bandwidth. New instructions enhance floating-point bandwidth, square root, and data conversion. Since POWER2 processors support a superset of the original POWER user-level instruction set, existing POWER binaries can run unmodified on the faster systems.

There are three categories for performance gains on the new systems. First, existing POWER binaries receive the performance benefits of the larger data cache, higher clock rates, improved bandwidth, and additional functional units. Second, recompiling can maintain portability while providing gains from recently added compiler transformations. For example, unrolling—the replication of an inner loop body—often exposes additional instruction-level parallelism. This process improves performance without sacrificing portability. Finally, when portability is not required, a recompiled application can also exploit new instructions. These improvements, along with the increase of the processor frequency to 71.5 MHz, boost the LINPACK performance to 140.3. This surpasses all workstation competitors and elevates POWER2 performance to supercomputer levels.

While both POWER and POWER2 systems extract maximum performance from a given technology, PowerPC systems reduce cost by exploiting single-chip capabilities. The ability to maintain compatibility across a broad spectrum of single-chip and multichip implementations is an integral component of IBM's "palmtops to teraFLOPs" strategy. Using the POWER Architecture as a base, developers of the PowerPC Architecture focused on changes that would allow more aggressive superscalar implementations, allow higher clock rates, and require less silicon. The PowerPC 601, the first implementation of the PowerPC Architecture, provides a "bridge" for application developers by implementing most of the instructions that were eliminated from the POWER Architecture base. Emulation of the missing instructions allows existing POWER binaries to run on the first PowerPC-based system—the Model 250. The PowerPC 601 pipeline organization is similar to that of the POWER processor. The branch unit has been enhanced, the integer pipeline is similar, and the floating-point pipeline has been optimized for single precision to minimize silicon area. In addition to enhanced scheduling, recompiling provides access to the instructions added to the PowerPC Architecture; the new single-precision floating-point arithmetics often provide the most noticeable performance effect.



Steven W. White



John R. Reysa

Targeting a Specific Platform

By paying careful attention during the definition of the architectures, the designers created an "intersection" set of instructions that are common across all three types of machines. This set is robust enough to provide nearly optimal performance. The default compiler setting selects instructions from this set. Programmer-specified compiler options (`-qarch`) can override this setting, allowing the compiler to target a specific implementation. In this case, compatibility is lost; however, additional performance can be achieved. For example, using POWER2-specific instructions destroys portability but provides significant performance gains on some applications, primarily those that are constrained by floating-point load and store instructions or those that are dominated by square root operations. Unlike the POWER and POWER2 systems that perform double-precision floating-point operations, the PowerPC Architecture also includes corresponding single-precision instructions. Specifically targeting the PowerPC 601 allows the compiler to exploit the faster single-precision operations and avoid special rounding operations that are required in POWER to truncate double-precision results to the single-precision format.

designers created "intersection" set instructions that common across three types of chines.

This interaction between compatibility and performance has been examined for two of the SPEC™ benchmark suites. For the integer suite, targeting a specific processor gains less than 1% over a binary that used only the "common" instructions. For the floating-point suite, targeting the POWER2 and PowerPC implementations provides roughly a 10% gain by exploiting the new POWER2 floating-point instructions and the PowerPC single-precision floating-point instructions. These results strongly suggest that all applications should be compiled with the default "common" mode. This is because targeting a specific implementation destroys compatibility and often provides only a modest gain. When performance is a primary concern and portability is not required, consider implementation-specific options for floating-point applications, especially for single-precision codes on PowerPC systems.

Performance with Portability

The small gains obtained by targeting a specific platform are dwarfed by opportunities from other compiler tuning options that maintain compatibility. Although applications are often compiled without optimization, simply adding the `-O` option dramatically boosts performance. This basic optimization level provides performance gains ranging from 26% to 508% on the applications in two SPEC suites. It provides roughly 70% of the performance of compiling the same applications with the best known compiler and preprocessor options. The remaining gains are obtained primarily through preprocessor- or application-specific options that also maintain compatibility.

Some of these optimizations benefit a particular platform while maintaining compatibility and performance on the alternative systems. An example is unrolling, which often increases the amount of independent work within an iteration. To avoid pipeline stalls caused by dependencies, the POWER2 design requires twice the amount of work between dependent operations compared to the single-pipeline POWER functional units. Since the PowerPC 601 floating-point pipeline is roughly twice as many stages as the POWER and POWER2 pipelines, unrolling also benefits the PowerPC 601. In cases where the previous compiler was too conservative with loop unrolling, this additional unrolling often benefits POWER systems.

In other cases, optimizations that benefit a particular platform maintain compatibility but degrade performance on the alternative systems. *Inlining*, which replaces a call to a routine with the inline equivalent of the routine, often provides performance gains. It frequently eliminates saving and restoring registers, allows optimization of the routine at the specific call site, and allows scheduling across the call boundary. Since a primary disadvantage of inlining is code expansion, inlining candidates are usually small routines that are frequently called from only a few sites. Experience has shown substantial performance gains for aggressive inlining on the POWER and POWER2 systems, both of which provide 32 KB instruction caches. However, the same degree of inlining on a PowerPC 601 system may be detrimental to performance. In some cases, the original instruction working set fits in its unified 32 KB cache and inlining results in thrashing of instructions and data. Similarly, trade-offs associated with transforming a single-precision application into a double-precision version vary across implementations because of differences in the cache organizations.

Strategy for Selecting Tuning Options

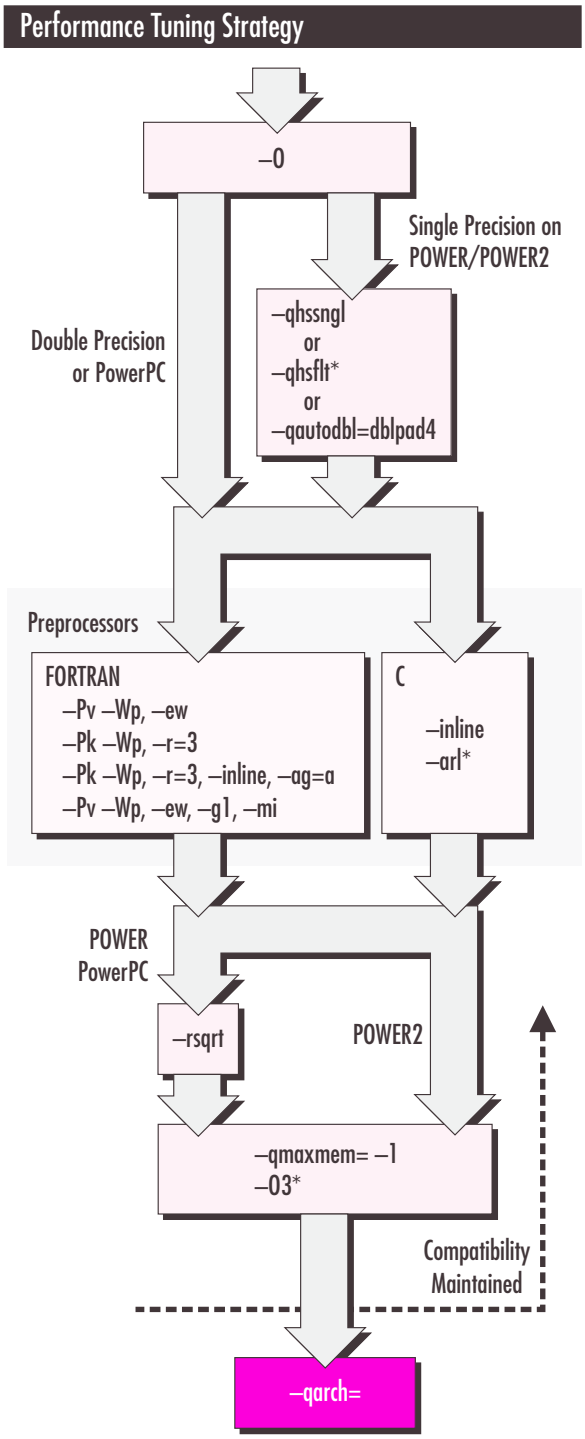
While the optimal set of tuning options is application specific, Figure 1 shows a high-level strategy for tuning experimentation. While additional optimizations usually increase compile time, the basic `-O` option is undoubtedly the most worthwhile performance option. Single-precision appli-

cations on POWER and POWER2 systems often benefit from minimizing the amount of round-to-single operations with the `-qhssngl` or `-qhsflt` options. Alternatively, an application can be transformed into a double-precision version (`-qautodbl`) however, while the rounding operations are eliminated, the data storage requirements increase.

Figure 1 shows several general-interest preprocessor options. Various combinations of these options are often beneficial. Application-specific information, such as inline candidates identified through profiling, can further refine the preprocessor option selection. For applications that are dominated by square root operations and run on platforms that do not support hardware square root, `-rsqrt` has shown gains. Other general tuning options allow the compiler to use more memory during code optimization (`-qmaxmem`) and request more aggressive optimization (`-O3`). Finally, if additional performance is required and portability is not essential, a specific implementation can be targeted (`-qarch`). Additional information can be found in the references listed below.

References

- *AIX XL FORTRAN Compiler/6000 User's Guide Version 3 Release 1* (SC09-1610).
- *IBM RISC System/6000 Technology* (SA23-2619).
- *Optimization and Tuning Guide for FORTRAN, C, and C++* (SC09-1705).
- Reysa, John et al. "Migration and Compatibility." *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000.* Prentice-Hall. Forthcoming in 1994.
- Stewart, Kate. "Using the XL Compiler Options to Improve Application Performance." *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000.* Prentice-Hall. Forthcoming in 1994.



The * denotes options that require knowledge of the application to guarantee correctness.

Figure 1. Performance tuning strategy

Experience has shown substantial performance gains for aggressive inlining on the POWER and POWER2 systems.

***John R. Reysa**, IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Mr. Reysa is a development manager in the processor performance group. He joined the IBM RISC System/6000 Division to work on I/O adapters for the RISC System/6000 and started his current position in 1992. He has a BS in Electrical Engineering from Texas A&M University and an MS in Electrical Engineering from the University of Texas. Mr. Reysa has received an Outstanding Innovation Award.*

***Steven W. White**, IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Dr. White is a senior engineer in the processor performance group. He has*