

White Paper

The Advantages of IA-64 for Java* and Other Component-based Development Environments

Information for Software Developers

intel®

Component-based environments, such as Microsoft's COM+, Java* technology, and Enterprise Java Beans (EJB), are an increasingly popular development environment for enterprise and Internet applications. While many existing e-Business sites still rely upon CGI, Perl, or Active Server Pages, larger e-Business software development projects increasingly depend on Java-, EJB-, or COM+-based component execution platforms to provide application servers for component-based applications. These application servers typically connect the Web server with back-end database servers. They contain the business logic for extracting information from the databases and leave the attractive presentation of its results to the Web server. The features of the IA-64 architecture and its first microprocessor implementation, the Intel® Itanium™ microprocessor, prove particularly useful for accelerating the performance of component-based code.

The Challenge

The performance penalties incurred by adding abstraction layers represent one of the most significant drawbacks of Java applications and of component-based applications in general. Specifically, the EJB and COM+ object-oriented approach involves the frequent invocation of small method calls. In the case of Java technology, the type-safety feature requires dynamic bounds-checking, null-checking, and exception handling. The resulting branch-heavy code performs particularly poorly on traditional architectures, whose execution units often come to a halt when encountering or mis-predicting branches.

Table of Contents

The Challenge	2
IA-64 Architecture's Contribution to the Solution	2
Summary	4

One optimization expert summarized some facts regarding the branch-heavy nature of Java code:

In Java, the average size of a method is only 68 bytes of Java bytecode. Further, approximately one in four instructions is a null or bounds check. ... Unlike C++, where methods are by default non-virtual, 69 percent of Java method calls are virtual. In fact, multiple levels of indirection may be required simply to obtain the location of the called method. (For finer-grained detail on Java performance benefits, see Carol Thompson's article at <http://www.performance-computing.com/features/9810f3.shtm>, which cites in turn an internal HP paper by M. Mehta.)

In other words, the frequent tasks of both following "pointer chains" caused by method calls and following branching code caused by bounds checking yields poor Java performance on traditional architectures. As with Java and EJB technologies, frequent method calls in COM+ present similar potential performance problems.

IA-64 Architecture's Contribution to the Solution

The four IA-64 architecture features most useful to overcoming Java technology performance barriers are predication, speculation, instruction parallelism, and the large number of registers. The intersection

of Java technology and other IA-64 architecture features, such as the 64-bit arithmetic and the large support of memory, are also contributors to the solution. In general terms, benefits enabled by the IA-64 technology in the Itanium processor include:

- **Predication**—Predication is the conditional execution of instructions, which allows code to avoid using branch or "if" statements. Instead, the chip executes both paths of the branch at the same time as the "if" statement is being run and then discards the unwanted path once the results of the "if" statement have been determined. Predication pays off for code with heavy branching and focuses on the process of executing parallel code paths.

Since this data-dependent code can be more readily scheduled and executed in parallel than branching code, predication can reduce the mis-predication penalties of component-based applications branch-intensive code. Thus, the execution of normal "if" and "switch" statements can benefit from this predicate feature as can the branching caused by Java technology's bounds-checking.

- **Speculation**—Microprocessor clock rates have historically grown much faster than the speed and latency of

system memory, leading to a variety of branch prediction and caching techniques. The IA-64 architecture's speculative loads take this one step further, allowing the compiler to schedule pre-fetching speculative loads from memory well in advance of the need for the data, thus removing the latency of the load operations and reducing processor stalls. Many competing RISC microprocessors can pre-fetch only after the last code branch, but the IA-64 architecture does not impose this limitation. This additional speculation capability pays off for code with heavy branching (due to "if" statements or procedure calls) and memory access. The IA-64 architecture also supports multi-way branching, in which up to 3 conditional branch instructions are evaluated at the same time. The result is reduced memory latency and faster application processing. Speculation allows the references to be fetched at an earlier stage of code execution, avoiding latency of waiting for memory loads after each reference or branch.

Thus the address locations and data needed for Java technology's bounds and null checks can be pre-fetched and that code can be executed without stalling. As a result, Java technology's bounds and null checks, which occur with each object reference, face a smaller performance penalty. The multi-way branching feature can improve performance or component-based applications' branch-intensive code. Similarly, method calls frequently

involve a chain of address references terminating with a branch to a particular address.

- Increased instruction parallelism—IA-64 compilers package instructions in bundles (three instructions per bundle) that can be executed in parallel; the Itanium processor provides multiple execution units to execute these instructions concurrently. Increased instructions per clock will benefit a broad range of software routines, particularly code that contains few operations which depend on data processed in the preceding few instructions.

For component-based environments, parallelism allows C/C++ compilers, or in the case of Java, a Java Just-In-Time compiler, to identify dependencies among a serial set of instructions and to explicitly schedule as many of those instructions as possible to run in parallel. This parallelism can improve Java, EJB, COM+ applications to execute faster by enabling multiple method calls to be executed simultaneously.

- Large 64-bit address space—Traditional 32-bit architectures only support up to 4GB of physical memory, often, 2GB of that is reserved for the operating system kernel. The IA-64 architecture supports a large, flat address space and physical memory addressability that will enable application performance for directory services.

Both Java and COM+ technologies provide the ability to persistently

store objects in memory. While not widely used today, a large persistent data store can provide a runtime environment that avoids disk access overhead for server environments with large amounts of physical memory. The performance of any application calling such persistent objects using a larger 64-bit memory address space will therefore improve.

- Very large set of registers—For about a decade, the IA-32 architecture has relied on eight registers (temporarily holding places for variables used by an application). Code with heavy data manipulation routines often require more than eight short-term holding places to avoid excessive and unproductive loads and stores. This overhead in moves and stores between registers and memory increases the likelihood for processor stalls and reduces performance.

The Itanium processor provides 128 integer, 128 floating point and 8 branch registers. The large register set enables fast access to data values, without having to access memory. With the Itanium process, not only can a single software routine address more registers without stalling, but multiple small methods also become far more likely to run efficiently in parallel without contending for limited register resources.

- Long 64-Bit Integers—IA-64 architecture supports 64-bit integer arithmetic, increasing support for code.

With support for 64-bit integer arithmetic, the IA-64 architecture will offer hardware support for Java technology's 64-bit integer long data type. Also worth noting is the fact that Java virtual machine implementations refer to remote Java objects using a unique 64-bit object handle known as the ObjID. Processing this handle on a 32-bit platform obviously requires at least twice as many instructions, making 64-bit hardware a more efficient platform for running server-side Java applications that use the handle while employing the Java Remote Method Invocation (RMI).

Summary

The frequent method calls of Java, EJB, and COM+ technologies represent a ripe target for acceleration by compiler techniques that utilize the Itanium processor's predication and speculation features. The hardware technique of predication also maps very well to rapidly handling Java technology-mandated bounds-checking and should alleviate Java technology's current performance penalties for that task significantly. By allowing compilers to convey more information about inter-code dependencies to the hardware, the IA-64 architecture's approach to parallelism should yield performance on a variety of code. The combination of these capabilities should make the Itanium processor an attractive platform for component-based environments.

Version 1.0 10/19/99

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

The Itanium processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.



For more information about Intel, please visit:
www.intel.com/ebusiness