The Advantages of IA-64 for In-Memory Database Applications

Information for Software Developers and IT Managers

Version 1.0 1/04/00

The Advantages of IA-64 for In-Memory Database Applications

January 2000

Software developers face the need to write more applications that make use of large databases under strict performance requirements. Real-time access to large databases, applications in large data warehouses and data marts, applications that access and manipulate on-line databases, and applications that can swiftly deliver information-rich web pages are but a few of their tasks. These applications must process information and access data quickly for e-Business transactions or telecommunication usage, such as cell phone and packet switching.

A major challenge to providing high performance access to database information is the time it takes to access disk drives. When disk access is required, disk access times add what can be an intolerable delay to efficient information access and utilization. Access to disk is typically hundreds to thousands times slower than access to memory.

Today, the disk access time challenge can be overcome. The price of random access memory has come down to affordable levels for many systems. This price reduction means that an entire database can be stored in system memory if the system processors can provide a very large linear address space.

A processor that supports a 64-bit address space may provide access to in-memory databases that range from tens of Gigabytes to thousands of Terabytes. In contrast, traditional 32-bit processors most often only address a maximum of 4 Gigabytes. To deliver the performance and response time needed for successful e-Business or telecommunications, software developers need to write applications for microprocessors that have 64-bit addressing capability. However, for best performance, applications need more than just a large address space.

More than just 64-bit Addressing

Several companies promote their 64-bit addressing capability for their existing processors. However, these processors lack the additional state-of-the-art IA-64 architectural features that enable Intel's first IA-64 processor, the Intel® Itanium™ processor, to provide peak performance and scalability for in-memory database applications. In addition to its 64-bit addressing capability, the Intel IA-64 architecture-based microprocessors provide additional architectural features for large in-memory databases such as a large number of integer registers, instruction set parallelism, predication, control and data speculation, and prefetch capability.

These features are summarized in Table 1.

Table 1: IA-64 Architecture Features

IA-64 Architecture Feature	**In-Memory Database Application Benefit**
64-bit addressing	Tens of Gigabytes to thousands of Terabytes stored in nanosecond access main memory eliminates millisecond disk access times thus improving application response time.
Large number of Registers and innovative register model	Data and intermediate calculations stored in on-chip registers reduce the repetitive load and store of intermediate data values thus improving the response time of an application's database request.
Instruction set Parallelism	Ability to execute instructions in parallel allows quick access simultaneously and manipulation of data derived from multiple rows and columns of a large in-memory database table or tables.
Predication	Predication allows the conditional execution of instructions before it is known whether the execution is needed. Predication allows more code to execute in parallel, the performance penalty of branch-dependent code is less, and applications with heavy branching speed up.
Control/Data Speculation	Control speculation allows certain load instructions to be scheduled before conditional branch instructions, rather than after. Data speculation is similar to control speculation but allow loads to be scheduled above stores. Both allow a reduction in the CPU wait states generated by branch-intensive code with high latency RAM accesses thus speeding application performance.
Instruction/Data Prefetch	Instruction prefetches can be signaled on branch instructions. Data can be prefetched with explicit prefetch instructions. Both prefetches speed application performance by reducing wait states.

Large Number of Integer Registers

Intel designed the Itanium processor to support 128 integer, 128 floating point, 8 branch and 64 predicate registers (for comparison, IA-32 processors support 8 registers and other RISC processors support 32 registers). The use of these registers allow more database data and intermediate calculations to be stored in on-chip registers and reduces the repetitive load/store of intermediate data values. These capabilities combine to greatly improve the overall response time of an application's database manipulation request.

The IA-64 architecture's registers provide great flexibility compared to registers built along traditional register model lines since they can be both "rotated" and "framed" registers. Rotating registers enable the efficient loop execution important to many in-memory database applications. They achieve the equivalent of software pipelining so that many loop iterations execute simultaneously with reduced cache misses; thus increasing application performance. Register frames easily handle the call-intensive code found in many database (in-memory and otherwise) applications. Moreover, they allow variable size blocks to be set up in the registers minimizing performance penalties due to excessive register save/restore operations and waste fewer register resources.

Instruction Set Parallelism

Traditional hardware-based instruction execution scheduling cannot efficiently utilize processor resources. With the IA-64 architecture's explicit instruction set parallelism, an IA-64 compiler produces parallel machine code that tells the processor how to execute more code in parallel.

This ability to execute instructions in parallel allows the IA-64 architecture to simultaneously access and manipulate the data derived from multiple rows and columns of a large in-memory database table or tables (multi-million row tables are common in certain applications). To best take advantage of this kind of explicit instruction parallelism, the IA-64 architecture provides a large number of registers.

To accommodate the high level of instruction parallelism, there are multiple replicated microprocessor functional units (e.g., for integer or floating point calculations). These work to execute as many instructions in parallel as possible and this, in turn, significantly increases in-memory database application performance.

Predication and Speculation

The predication and speculation features complement each other to improve in-memory database performance. The predication feature allows the scheduling of instructions before it is known whether or not the execution is needed. The value of a predicate bit (associated with each instruction), stored in one of the IA-64 architecture's 64 1-bit predicate registers, determines whether the instruction is executed.

The predicate bit, assigned by the compiler, acts like a switch. If the predicate is true, the instruction is executed. Predicate bits can be set as the result of Compare or Logical instructions. As a result, many branches are removed and "branch mispredicts" significantly reduced.

Since predication allows more code to be executed in parallel, the performance penalty of branch-dependent code is less. In-memory database applications with varying or non-predictable workloads or queries (e.g., data mining applications), and the subsequent heavy branching, will realize the immediate benefits of the IA-64 predicate feature.

Since processor speeds are far in excess of RAM access speeds, current processors often "stall" waiting for new code or data to be retrieved. Speculation, or pre-fetching code, overcomes a major reason why the instruction pipelines of current microprocessors are not full. In contrast to well ordered, repetitive calculation-based code, commercial database applications typically have heavy branching. So, as is the case with predication, database applications will benefit greatly from the IA-64 speculation feature.

Control speculation allows certain load instructions to be scheduled before conditional branch instructions, rather than after. In most existing architectures, load instructions typically cannot be moved ahead of conditional branch instructions due to the possibility that the load instruction may not be executed at all after the branch.

If moved ahead of the conditional branch, the load instruction may cause an exception to occur. For this reason, conditional branch instructions often act as an instruction scheduling “barrier” to today’s compiler. However, it is advantageous to break this barrier and move the load instruction ahead of the conditional branch if there is a high probability that the load instruction will be executed.

To implement control speculation, the IA-64 architecture allows a load instruction to be implemented as two separate instructions — a speculation load instruction and a speculation check instruction. A speculation load instruction can be placed by the compiler before the conditional branch. At the original site of the load instruction after the conditional branch instruction, the compiler places a speculation check instruction. This speculation load allows the reduction of the number of CPU wait states generated by branch-intensive code with many high latency RAM accesses.

With a high probability that the load instruction moved ahead of the conditional branch instruction will not cause an exception, the speculation check will typically note that no exception needs to be raised, and instruction execution can continue. The in-memory database application has the advantage that the data from the load instruction is now present in a register.

The IA-64 architecture also provides data speculation. It is similar to control speculation but allows loads to be scheduled above stores. Finally, the IA-64 architecture provides a prefetch capability. Instruction prefetches can be signaled on branch instructions. Data can be prefetched with explicit prefetch instructions. Here too, wait states are reduced, yielding faster performance.

Summary

With the growth of e-Business and larger internet and telecommunications infrastructures, the need for faster database access is becoming more important. IA-64 architecture-based microprocessors offer the 64-bit addressing capability, large number of registers, inherent instruction level parallelism, predication, control/data speculation, and prefetch capability needed to optimize the performance of in-memory database applications for many e-Business solutions. As a result, software developers who seek competitive advantage for their applications will develop code optimized for the Intel Itanium processor, Intel’s first IA-64 architecture implementation.