Multicore CPUs for the Masses

Mache Creeger, Emergent Technology Associates

Will increased CPU bandwidth translate into usable desktop performance?

Multicore is the new hot topic in the latest round of CPUs from Intel, AMD, Sun, etc. With clock speed increases becoming more and more difficult to achieve, vendors have turned to multicore CPUs as the best way to gain additional performance. Customers are excited about the promise of more performance through parallel processors for the same real estate investment.

For a handful of popular server-based enterprise applications, that may be true, but for desktop applications I wouldn't depend on that promise being fulfilled anytime soon. The expectation for multicore CPUs on the desktop is to have all our desktop applications fully using all the processor cores on the chip. Each application would gracefully increase its performance as more and more processors became available for use. Just like past increases in clock speed and application bandwidth, increasing the number of processor cores should produce similar performance enhancements. It works for the popular enterprise applications, so why not for desktop applications? Sounds reasonable, right? Don't count on it.

Sure, the major enterprise applications such as Oracle, WebLogic, DB2, and Apache are designed to take full advantage of multiple processors and are architected to be MT (multithreaded). They have to be for the large SMP (symmetric multiprocessing) servers that are the meat and potatoes of their market.

Even though the concept of using concurrent CPUs to increase overall software performance has been around for at least 35 years, remarkably little in the way of development tools has made it to the commercial marketplace. As a result, the vast majority of applications are single-threaded. Although multicore CPUs will allow you to share a mix of applications across multiple processors, individual application performance will remain bounded by the speed of an individual processor. Application performance will remain the same regardless of whether you have one or 100 processors because each application can run on only one processor at any given time.

With the possible exception of Java, there are no widely used commercial development languages with MT extensions. Realistically, until now there has not been much of a need. The widespread availability of commercial SMP systems did not really arrive until the early 1990s, and even then multithreaded applications came slowly.

When I was at Sun, the company rewrote SunOS to take advantage of its new multithreading architecture. It was a long and painful process. Initially, subsystems were rewritten with locks at either end so they would be assured to run as one big single thread (MT-safe) and then rewritten again to be fully MT optimized (MT-hot) for maximal concurrency. Everything was designed by hand and there were no tools to manage the complexity.

Around the same time, Sun implemented a set of user MT libraries that applications could use. As larger SMP servers started to appear on Sun's roadmap, the major enterprise application vendors saw that they too had to make the investment in converting their software to MT. The experience was equally painful and similar to the SunOS MT rewrite. Recognizing the need to make these applications run MT-hot in order to sell their new SMP servers, Sun leveraged its experience by assigning engineers to these companies to help them in their migration.

The situation today is quickly becoming a replay of what happened 10 years ago. Application vendors requiring more CPU bandwidth can no longer count on increased clock speeds for better performance and functionality. Most large-scale client-side applications are written in C or C++ and historically have been designed to be single-threaded. Making applications MT-hot is still a labor-intensive redelivery process. Although a few vendors, most notably in the multimedia area, have made some MT enhancements to their applications, they have just started to pick off the low-hanging fruit. With multicore CPUs, widespread desktop performance and functionality improvements are still years away.

What have the development tool vendors been doing as MT architectures have evolved during the past decade or so? It's not as if anyone in the computer industry did not see this coming. What can we expect in the future? Given where the industry is today, the introduction of

Continued on page 63

Continued from page 64
multicore CPU-based desktop systems will stall as customers figure out that most of their applications run no faster on a dual- or quad-core system than on a unicore system. To sell more machines/CPUs, hardware vendors will have to do what Sun did and "encourage" application vendors to redesign their applications to be MT-hot. Desktop application vendors who have been able to depend on continual CPU clock increases will now have to invest in a long and painful rewrite of their software to gain the next jump in performance and functionality. All this could take years. Moreover, more agile companies will now have an opening to make MT-hot investments faster, potentially snagging customers from incumbent vendors that are too slow to make the transition.

What is frustrating is that all of this could have been avoided. MT has been on the horizon for at least a decade. Because technology companies take a myopic quarter-by-quarter view in their planning, they missed the bigger trend of multicore CPUs and their implications for the desktop. As a result, the tools for MT development are not in place as these new CPUs hit the market. With the exception of Java's minimal MT support, things look fairly close to what the large enterprise application developers had to work with more than 10 years ago.

Sadly, I see the following scenario playing out. It will take several years of pain for application developers to rewrite their code to be MT-hot. Once a methodology for conversion has been established, IDE tool vendors will start bringing out automation extensions that help manage MT development complexity. These two processes could easily take three to five years. Once MT-enhanced IDE products become established, language extensions will follow. A commercially accepted development language with fully integrated MT control structures should come into widespread use in five to seven years. In the meantime, don't count on that instant performance increase for desktop applications with the release of each new CPU family. With multicore systems, having CPU bandwidth on the desktop and being able to use it are going to be two very different things. Q

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

MACHE CREEGER (mache@creeger.com) is a 30-year veteran of the technology industry. He is the principal of Emergent Technology Associates, marketing and business development consultants to technology companies.