



Commentary

Multithreaded, Multi-core Processors: Hey, Where Are the Programs?

Executive Summary

Simply stated, in IT a “thread” is a set of application instructions. Multiple threads, therefore, are multiple sets of instructions and a multi-threading processor or server is one capable of executing multiple instruction sets.

For almost a decade, graphics processors and commercial microprocessors have been capable of executing multiple threads per clock cycle. And more recently, with the advent of multi-core processors like IBM’s Cell, these processors can now execute multiple threads across multiple, tightly-coupled cores (central processing units closely linked on the same die).

The best way to exploit these new multi-core processors for maximum performance is to constantly feed them application instructions (threads) that can be executed in parallel. For example, an application that consists of multiple threads can be parsed (broken into discrete components), and each component can then be handed off to a separate, discrete processors on the same core for simultaneous processing. In this manner, the processing of application threads is handled in parallel — and application processing, therefore, is greatly accelerated (in some cases, applications can execute ten to a hundred times faster using this approach).

To gain maximum advantages from multi-core, multi-threaded computing models, a framework and logic needs to be implemented to efficiently manage the processing of threads — and to ensure that all threads combine after processing to produce the expected result.

This is not as easy as some make it sound. When it comes to implementing such a framework, application developers generally have three choices:

1. Cobble together application libraries, debuggers, and toolkits from various sources (including microprocessor makers and the open source community);
2. Buy an integrated framework; or,
3. Do nothing (continue building single threaded applications that execute in a single processor).

Today, most application developers are choosing option #3.

Multithreaded, Multi-core Processors: Hey, Where Are the Programs?

Fixing This Situation

It seems a real shame to waste multi-threaded, multi-core processing resources by not building applications that are designed to exploit them. But, in my opinion, not all applications really need to be parallelized to exploit multi-core, multi-threading architectures. In fact, many commercial business applications have been designed to process work linearly — so breaking applications apart into discrete components that can then be processed in parallel may actually provide little benefit (and doing this may actually increase application development overhead prices).

However, there are classes of applications that benefit in extreme ways from the ability to exploit parallelism across multi-core, multi-threaded processors. Three-dimensional (3-D), graphically driven applications written to process and render images are probably the most prominent example. In fact, specialized processors (graphical processing units — GPUs) have been built to accelerate processing in these environments. Sony, who co-developed the Cell processor with IBM and Toshiba is a good example of this approach. And vendors who build 3-D graphic games have even devised their own toolkits and libraries to exploit GPU multi-core performance.

Recently, a company that I wrote about last year — RapidMind — released a multi-core development platform for implementing multi-threaded applications on commercial Intel and AMD multi-core x86-based processors. As a result, x86-focused developers looking for an integrated framework (and a knowledgeable support organization) have a place to turn for pre-integrated multi-thread/multi-core tools and libraries. Note: RapidMind's platform also supports Cell processors and GPUs.

The Benefits in a Nutshell: What a RapidMind Customer Told Me

In the past I have found it is very difficult to get users of advanced tool/framework environments to conduct interviews — mostly because early adopters are reaping such competitive advantages that they are loathe to let their competition in on the secret. However, I recently got lucky in this respect and had the opportunity to speak with Sudy Shen, the president and CEO of the Masstech Group — a global developer of media lifecycle management, workflow productivity enhancement, business continuity and compliance that enables the management, conversion, distribution, and sharing of digital content across IT storage and networking infrastructures.

Shen's company makes use of RapidMind's multi-core development platform to help build transcode digital content and present that content in the forms its customers want. For instance, a digital movie could be transcoded and transmitted almost instantaneously to a user who wished to view that content in high-definition format. Or that same digital movie could be transcoded and rapidly presented in a lesser resolution to a mobile device user. Using this model, users pay not only for the content, but also the quality that they desire.

Shen told me that his company adopted RapidMind for several reasons including:

Multithreaded, Multi-core Processors: Hey, Where Are the Programs?

1. It took Masstech out of the business of writing tools, collecting debuggers, and so on — and let them concentrate on what they do best (their core content business);
2. The company (RapidMind) had deep expertise in building multi-core, multi-threaded applications — so his developers could exploit that expertise in order to get products to market more quickly; and,
3. Shen especially liked the company's migration path across GPUs, CPUs, and Cell processors (because his business necessitates moving across multiple device types).

As for the results that he is seeing by effectively exploiting multi-threaded, multi-core architecture, Shen qualified the performance of his applications after using the RapidMind multi-core development platform as “better than real-time.”

One other point that Shen raised was that, in his business, building transcoders has often relied on using specialized hardware that does not sell in volume — hence, encoding/decoding systems have typically carried a very heavy price tag. Using RapidMind to develop solutions on industry standard platforms should, he believes, both reduce hardware acquisition costs and also increase his market share (as more customers will be able to afford Masstech products).

Summary Observation

Not all applications are candidates for parallelization. But for those that are, the payback on parallelizing multiple threads over multi-core architecture can be stupendous in terms of both performance and broadened market appeal.

Masstech's transcoding example is but one of many application environments that can benefit immensely from the platform/deep expertise that RapidMind offers. And RapidMind is working with financial, government and scientific customers who are also seeing major benefit from enhanced multi-threading/multi-core development and performance. Most of these customers are prohibited by company policy or governmental regulations from advocating the RapidMind approach — or do not wish to share the competitive advantages that they are achieving. But rest assured, they're out there — and RapidMind can provide general details about the results customers have achieved with the company's multi-core development platform.

Clabby Analytics
<http://www.clabbyanalytics.com>
Telephone: 001 (207) 846-0498

© 2008 Clabby Analytics
All rights reserved
April, 2008

Clabby Analytics is an independent technology research and analysis organization that specializes in information infrastructure and business process integration/management. Other research and analysis conducted by Clabby Analytics can be found at: www.valleyviewventures.com.