JOIM
www.joim.com

JOURNAL OF INVESTMENT MANAGEMENT, Vol. 4, No. 2, (2006), pp. 76–81

TECHNOLOGY REVIEW

MULTIPLE-CORE PROCESSORS FOR FINANCE APPLICATIONS

Sanjiv R. Das[a]

1 Introduction

Large chip-makers are on the threshold of a mass-market distribution of multiple-core processors. Multliple core processors embed more than one sub-CPU on the main chip, each one being denoted a "core." This note discusses the role of this technology for applications in the financial services industry. Figure 1 shows the 64-bit AMD Athlon dual-core processor. Figure 2 shows the die for the Intel Extreme Pentium dual-core processor.

Figure 1 Dual core processor (top and bottom).

Apple Computer reports that the speed-up of creative applications like video-editing, multimedia, etc. is in the range of 40–70% with dual-core versus single-core processors. For mathematical and scientific applications the performance improves by about 20–75%. Intel reports that it plans to ship more dual-core processors than single core in 2006 (Reuters, November 2, 2005). The technology is advancing rapidly as well, and is moving to 65 nm designs from 90 nm, allowing more computing power to be squeezed onto the same silicon real estate. These are expected to decline to the 32 and 22 nm scale. (A nanometer equals 10^{-9} of a meter.) Multi-core processors were envisaged almost two decades ago at Intel by Gelsinger *et al.* (1989). Hyper-threading technology presaged the multi-core approach, in that it provided multiple virtual processors; with the advent of multi-cores, we are able to boost the gains from hyper-threading (see Thulasiram and Thulasiram, 2003, for an application in finance) by enhancing virtual processors with physical ones, providing multiplicative performance improvements (Think "hard-core" finance).

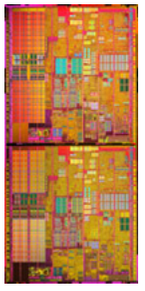

Figure 2 Dual core Extreme die.

[a]Department of Finance, Santa Clara University.

2 Financial application taxonomy

These advances in technology are likely to have an impact on the finance industry, which is incessantly hungry for faster technology. Multi-core processing is a special case of distributed processing, leading to theoretical connections to social and economic structures (see Huberman, 1996). A simple framework for a two-way classification of applications in finance is as follows:

1. High CPU, low I/O: these are applications that require complex calculations and vast amounts of numerical analysis, but have very few data inputs and correspondingly, very low I/O.
2. Low CPU, high I/O: the opposite. We present below examples of both these types of activity in financial markets.

The first type of application is better suited to multiple core technology. While the framework suggests a simple bifurcation of technology needs for the finance sector, it hides a somewhat richer categorization.

Financial services may be classified into the following categories:

1. *Market microstructure businesses*: Auctions (EBay), or market making, such as the New York Stock Exchange (NYSE), American Stock Exchange (AMEX), NASDAQ, etc. In this sector, the need for real-time data processing and handling large volumes of data is paramount.
2. *Derivatives markets*: CBOT (Chicago Board of Trade), CBOE, etc. These are also related to the realm of market microstructure, but are distinguished from the previous category in that the need for mathematical algorithms is substantially higher. Derivatives markets are becoming increasingly defined by a growing complexity of mathematical analysis, leading to security prices based on solutions to problems that need very high CPU times.
3. *Risk management*: The extent of legal regulation has grown substantially, especially in more complex markets, in which inaccuracies in assessment of portfolio risk have led to many financial blow-ups in the recent past. This activity usually involves the revaluation of entire portfolios within a few hours, tasks that are often outside the capabilities of current computers. This leads to inefficient approaches, such as breaking down a job and distributing it, making portfolio aggregation harder, and less failsafe. Most of these tasks are high CPU and high memory, and would be amenable to multiple-core solutions.
4. *Retail services*: Tasks such as online banking, which are high I/O tasks and less likely to be requiring multiple-core technology.
5. Personal financial services: This is a different class of retail service, also predominantly I/O driven, but becoming increasingly complex quantitatively. For instance, vendors who offer individuals abilities to trade complex derivatives (such as OptionsXpress) will be required to offer increasingly complex tools for pricing derivatives. These tools will need to be more general and not targeted (i.e., specialized) to any one type of product, thus requiring general algorithms (on the server side) that will need much more numerical analysis than previously anticipated.
6. *Back-offices of trading businesses*: Whereas a dealing room has the luxury of pricing and trading through the day, using multiple machines across trading desks, at the close of business, the back-office has to process and revalue all deals done in a few hours on a few computers. If the trades require complex mathematics, as they often do, the need for fast computation at the back office is more crucial than at the front office.
7. *Data mining*: Investment analysis now uses huge databases in a data mining approach to find profitable trades. This is mainly a high I/O task.

8. *Decision systems*: Financial houses employ automated decision systems in mainstream retail lines. For example, the processing of credit card applications is often conducted using a neural net. Most of these applications do not require CPU intensity, and are also mainly I/O driven.

Clearly, there are some applications that will lend themselves favorably to multiple-core processors, and there are many that will not. These chips are already present in high-end personal computers, and are not priced out of the range of most consumers, and definitely well within the cost levels of businesses.

3 Canonical applications

The pricing of derivative securities is a classic application in financial houses that calls for an exceedingly large number of CPU cycles. There is a graded level of processor effort.

1. Simple derivatives are easy to price, and this is often done using analytical (closed-form) equations, requiring very low processor speed. Such models run easily on spreadsheets.
2. Derivatives needing numerical solution. Even in the simplest option pricing situations (e.g., vanilla equity options), closed form solutions may become infeasible. For example, if we move from pricing an European option (exercisable only at maturity) to an American one (which may be exercised at any time up to maturity), we are forced to use a model that is no longer an equation, but a numerical calculation on a binary tree (or a multinomial tree for that matter). This change entails a huge jump in CPU cycles.
3. Derivatives needing fast integration. When we proceed to higher-dimension derivatives, CPU speed becomes even more important. In contrast to simple equity options, where uncertainty depends on a univariate stochastic process (i.e., the equity price), multidimensional options need to model vector stochastic processes and this leads to an exponential blow up in computational complexity. As an example, consider a longer term equity option also known as a LEAP. When an option's maturity extends out to 2 years, we may no longer assume that interest rates are constant over time. Hence, in addition to the stochastic process for the equity price, we also need to model the stochastic process for interest rates and the interaction of the two processes. If the option we wish to price is European, we may need to undertake a numerical integration over a bivariate probability distribution. While this is not a big problem nowadays, it is still a very time-consuming process when a portfolio comprises thousands of option contracts.
4. High-dimensional options of American type. If the options are American, then again, numerical integration is inapplicable, and we are forced to resort to building high-dimensional trees, on which computation must be carried out.

Sometimes, we get lucky, and are able to describe higher-dimensional processes on trees that are *recombining*, i.e., the number of nodes in the tree does not grow exponentially but only polynomially. But this is only possible under very restrictive assumptions that distort the stochastic process being embedded on the tree. Hence, for the solutions of the future, the derivatives industry is looking for a highly general solution to the problem. The most likely place where this will come from is in hardware, as a wide range of software solutions that have been explored so far, and further incremental gains may not amount to much (more on this later). Therefore, multiple-core processors are a very promising solution.

For the nonfinance computer technologist, further elucidation is worthwhile. As an example, consider the option pricing problem for a call option on Intel

stock. The core of the solution lies in making an assumption about how Intel's stock will move over the life (denoted T) of the option. This is commonly done by assuming that the return on the stock follows a properly parameterized Brownian motion. In our computer model, we may describe the evolution of the Brownian motion discretely over n periods, each of time interval $h = T/n$. The movement of the stock price is embedded on a binary tree, with an initial node S_1, leading to two nodes after time h, an up node S_2, and a down node S_3. In turn, starting from S_2, we may branch up to S_4, and down to S_5. Likewise, from S_2, we branch to S_6 and S_7. And so on. After n periods we will have 2^n leaves of the tree (we assume that the tree does not recombine in its evolution).

It is easy to show that the computational effort required to price the option is proportional to the number of nodes on the tree, and is, therefore, $O(2^n)$ (exponential). This leads to a computational blow up for large trees, as we shrink h (increase n) to come ever closer to an accurate approximation of the desired Brownian motion.

Of course, we have known now for about three decades how to work around this problem—we build trees that reconnect, so that we go from S_1 to S_2 and S_3. Then from S_2 to S_4 and S_5. Now from S_3 we branch to S_5 and S_6. In this case, we get a *recombining tree*, which has only $(n+1)(n+2)/2$ nodes, where computational effort is $O(n^2)$ (polynomial). But, recombination only works in very special cases of stochastic processes, and mostly traders have continued to use approximate (and possibly incorrect) stochastic processes simply because they are convenient. This reminds one of the parable of a man searching for his keys under the lamp post, because it is where the light is, not because it is where he lost his keys. Wall Street quants are well aware of this issue, and would welcome a fast solution to an exponential blow up tree that better represents the real stock price process rather than one that may be conveniently modeled using a recombining tree.

This problem worsens dramatically when the dimension (d) of the tree increases. Suppose we are pricing an option on the maximum of two stocks. We then need to model the movement of both stocks. This means that each node on our tree will have four branches emanating from it. Since each stock has two outcomes, up and down, the joint process for the stocks will have four outcomes, i.e., {{up,up},{up,down},{down,up},{down,down}}. Hence, for a *non-recombining* tree, the computational effort is $O(2^n d^n)$.

There are many such problems that firms are grappling with. The ability to price derivatives off a richer model will confer substantive trading gains to Wall Street firms, analogous to the benefits of a Formula One driver using a car orders of magnitude better than the competition. Leveraging better technology translates directly into competitive edge.

4 Current solutions

Many option pricing algorithms involve tasks that may be handled separately, i.e., broken down. For example, on the trees described above, the top half of the tree may be computed independently of the bottom half of the tree. This opens up possibilities for many known technologies in obvious ways.

It may be useful to summarize what is currently being done in the absence of multiple-core approaches. I briefly identify four technical approaches used in the derivatives pricing industry. Each of these approaches is non-hardware based.

1. *Multi-threading*. The tree may be computed off multiple threads. Modern programming languages support this, and so it is easy to

implement. However, the extent of speed-up is limited.

2. *Parallelization*. The job is broken down into subparts and distributed across machines. This is popular for large portfolios. Each contract within the portfolio may be distributed to a machine, and then the results across contracts are aggregated on a central machine. This is a popular approach, used, for example, in the pricing of mortgage portfolios.
3. *Simulation*. Rather than compute all possible scenarios facing an option (i.e., traversing all nodes via all paths through the pricing tree), we may simulate a limited sample of paths through the tree only, and rely on a simulated estimator of the true value of the option. This is very popular. There are many firms that offer simulation software expressly targeted to such applications. There has also been a lot of research on these applications, and many books that summarize the research and practical implications of this work.
4. *Approximations*. One may think of three types of approximations in use.
   - One, where the solution to the numerical problem is replaced by an approximate analytical one. These solutions are hard to find, and are very specific, so do not result in general solutions with mass market appeal.
   - Two, using approximate recombining trees in place of an exact non-recombining one. Again, these are not general solutions.
   - Three, smaller (in n) trees which may not converge as desired to the true process being modeled. On top of these smaller trees error reduction techniques may be applied. These solutions may be somewhat more general, but are not apt either.

Therefore, much effort has been devoted to handling these problems. As the complexity of derivative securities grows, the need for speed will grow rapidly.

5 Solutions

The solutions we have seen so far over the past decade are really "soft" solutions, which have distorted the mathematics or are based on software approaches. What new chip technology offers is a way to mainstream "hard" solutions to these problems.

We may identify two categories of solutions (amongst the many more specific forms that exist) which are of immediate interest to the financial institution mass market:

1. Application-specific integrated circuits (ASICs)—there are many problems that are well defined, and have huge trading volumes. Specific chips to tackle these may be well worth the effort, certainly for Wall Street firms, and for many vendors. This would also include the class of floating point gate array (FPGA) systems.
2. Multiple-core processors—these provide a general solution to most problems, requiring only in-house Wall Street expertise in developing applications to leverage the capabilities of the device. This is mass market, as the target customers include all financial institutions and all vendors who provide support to the derivatives business worldwide.

While we have identified just one line of business (derivatives) there are many more that will progressively come on stream.

6 Summary

The categorization of finance applications described here provides a framework for matching computing technologies to financial problems. It is likely that we will see faster growth in computational finance

techniques. The prognosis is that innovative benefits will be greater on the hardware side than in software.

It is hard to forecast the market for dual-core processors from Wall Street. Whether they will first be deployed on the desktop or on the server side is also an open question, depending to some extent on the evolving client–server model in trading environments.

From a research point of view, many interesting problems arise. Understanding the trade-off between approximation approaches and exact computation on multi-core processors is a cost–benefit one at a first-cut, but on a longer term, the range of products that may be offered also depends critically on how rapidly the new technology is adopted. To this end, undertaking research that quantifies cost and efficiency gains needs to be done, but even more useful would be work leading to a broadening of the range of techniques that can take advantage of multi-core processors, such that a wider range of applications may be supported with the technology.

References

Gelsinger, P. P., Gargini, P. A., Parker, G. H., and Yu, A. Y. C. (1989). "Microprocessors, Circa 2000." *IEEE Spectrum*, October, 43–47.

Huberman, B. A. (1996). "Computation as Economics." Mimeo, Xerox Palo Alto Research Center.

Thulasiram, R. K. and Thulasiraman, P. (2003). "Performance Evaluation of a Multithreaded Fast Fourier Transform Algorithm for Derivative Pricing." *The Journal of Supercomputing* **26**, 43–58.