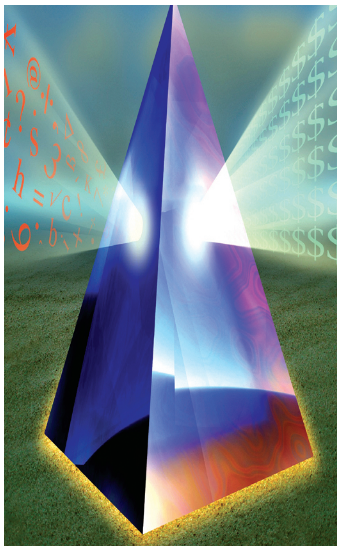
THE NEWSPAPER OF FINANCIAL ENGINEERING™

FINANCIAL ENGINEERING NEWS

Universal Coverage of Financial Innovation www.fenews.com



Where Numerics Matter

Thomas Coleman, Ph.D.

Fast Portfolio Computations Using Parallelism, Clusters and Web Services

The problems of financial engineering, and more generally quantitative finance, represent an important class of computationally intensive computing problems arising in industry. Many of these computation-intensive problems involve portfolio calculations. Examples include determining the fair value of a portfolio (of financial instruments), computing an effective portfolio hedging strategy, calculating the value-at-risk of a portfolio and determining an optimal rebalance of the portfolio. Because of the size of many practical portfolios, and the complexity of modern financial instruments, the computing time to solve these problems can take many hours, even days. However, with the movement of many organizations to automated (near) real-time systems, speed is of paramount importance. Days, even hours, may be impractical.

Moreover, the computing challenge increases as future "scenarios" are considered. For example, hedge fund managers wish to peek into the future. How will the value of my portfolio of convertibles change going forward if interest rates climb but the underlying declines and volatility increases? If the risk of default of a corporate bond issuer rises sharply over the next few years, how will my portfolio valuation be impacted? Can I visualize some of these dependencies and relationships evolving over the next few years? Within a range of parameter fluctuations, what is the worst case scenario? Clearly such "what if" questions can help a fund manager decide *today* on portfolio adjustments and hedging possibilities. However, peering into the future can be *very expensive*. Even modest "futuristic" questions can result in many hours of computing time on powerful workstations.

How is one to significantly speed up these calculations to allow for aggressive portfolio management? There are several possible roads to take. One ill-advised solution is to cheat: e.g., reduce the number of simulations (disregarding accuracy concerns), use very coarse approximations, cut corners, etc. Of course using coarser approximations can sometimes work in numerical work but accuracy implications must be carefully understood: There is no point in computing a dangerously wrong answer, fast or slow! A second approach is to exploit problem structure and apply tailored structured numerical techniques in the design of the solution – this can sometimes produce dramatic improvements in running times (a good example here is efficient frontier calculations in the context of a portfolio representation using factors. We will discuss this type of example in greater detail in a subsequent article.) Yet another alternative is to move the entire portfolio system to a supercomputer. This certainly can be effective though, unless a supercomputer is conveniently at hand, this approach can certainly be costly. It may also involve the use of more than one environment – moving between system environments can lead to errors as well as "real" time delays. This can be a cumbersome, inefficient and "user unfriendly" approach.

continued on next page

Exploitation of parallelism, in the setting of (commodity) clusters, provides another approach which can be quite effective, and relatively low-cost, for many of the compute-intensive problems that arise in finance. A cluster is just a collection of processors (or machines) in some way linked together, or grouped, so that together they can be orchestrated to solve a single problem. The word "commodity" is used to indicate that these need not be expensive specialized processors, but can be processors used in standard PCs or, in some cases, standard servers.

The use of clusters in scientific computing has been a significant growth area in the last few years – many of the largest machines in the world are really just clusters of processors linked with an interconnect (to allow communication). Indeed, many of the big Wall Street firms are using clusters in some computational contexts and currently exploring their use in others. Some of these efforts tie in with grid computing; this is the technology which locates appropriate computing resources, in a given universe of possible resources, and then makes those resources available for a given requested task. In many cases, though not always, the computing resources that are corralled for use form a commodity cluster.

Conveniently locating parallelism in a large-scale scientific computing task, and then exploiting this parallelism in a productive way, is generally not easy. In addition, a solution, once developed, is often not transportable and may have a short shelf life. The many challenges in parallel scientific computing include determining the appropriate level of granularity, determining a good trade-off between computational and communication concerns, effective data distribution, and coordination/synchronization of computing tasks. Add to this mix of concerns the reality of existing legacy code codes and the use of third-party software and you can feel a real headache coming on. It is no wonder that commercial and industry use of parallel computing has seriously lagged the use of this technology in university and government labs (where teams of Ph.D.s can often be dedicated to these tasks).

There is progress. Increasingly there are tools, public and commercial, to help with the effective parallel use of a cluster environment. Argonne National Labora-tories has a set of tools available to enhance grid, cluster and parallel computing (http://www-fp.mcs.anl.gov/division/software/). On the commercial side, companies such as Aspeed (http://www. aspeed.com/) and Data-synapse (http://www.datasynapse.com/) have commercially available products with particular applicability and experience in the financial services sector. There are also more basic free tools now available – both for communication and computation, for help in the design of parallel "cluster-friendly" computational finance procedures. Many basic building blocks are now available.

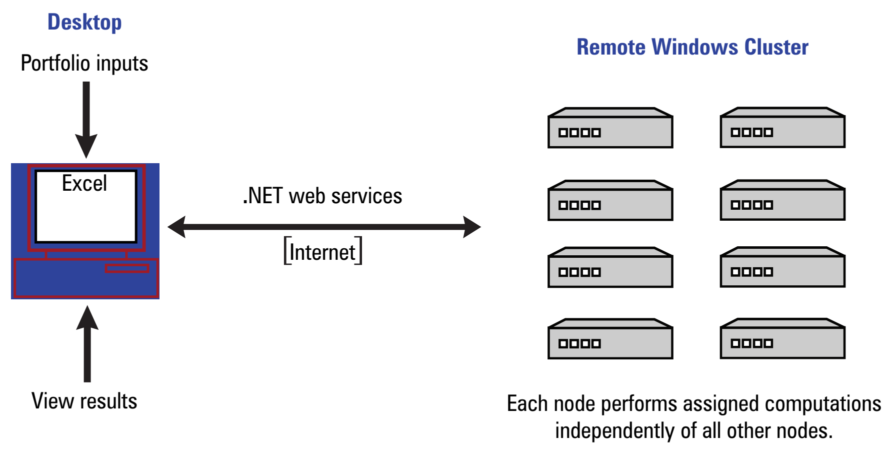
However, our thesis is that many of the computationally intensive problems that arise in finance are actually easier to solve using parallelism within a commodity cluster environment, than many of the more traditional large-scale scientific problems (e.g., design of an aircraft wing). The reason is that often the computing demands are severe primarily because the portfolios are large. This type of request often leads to a "loosely-coupled" computation, where the bulk of the work can be computed in large chunks with relatively little communication or coordination. Hence, a simple master-worker framework can be used. The master processor, perhaps the desktop machine, doles out work as it becomes available in good-sized chunks to processors (as they finish their previous tasks and are thus ready for more work).

Many portfolio evaluation problems, hedging problems and risk computations have exactly this flavor. And there is more good news. In this setting, i.e., when solving computationally intensive problems that are loosely coupled, it may be possible to design an effective parallel approach targeted to clusters without a large investment in hardware or specialized parallel tools and software. Indeed in our Manhattan center, the Cornell-Waterloo Solutions Lab (formerly CTC-Manhattan), we have illustrated that many significant finance problems can be effectively solved in an environment that is already common in many workplaces: a Web services cluster environment with computations driven straight from an Excel spreadsheet representation of a portfolio. A graphical representation of the environment we have in mind is given in Figure 1. A short description of a Web service is given in the appendix at the end of this article.

This can be particularly attractive in finance since an Excel representation of a portfolio is common. Note that a special interconnect, beyond Internet connectivity, is not needed and the cluster need not be local (and could be distributed in disperse regions of the world.) This is *not* a general solution for many of the large-scale problems that arise in the scientific computing community; it *is* a very effective solution of some of the common large-scale portfolio problems that arise in finance. Note also that while we have specifically worked with .Net and Windows clusters in our lab, Linux solutions

continued on next page

Figure 1. Desktop Access to Cluster Using Web Services



(perhaps with an Excel front-end) are also possible. What is required is that cluster processors are designated Web servers.

Table 1 lists some of the general computational finance (portfolio) problems that can be effectively solved in the Web services framework.

Table 1. Solving Computational Finance Problems on a Cluster Accessed Using Web Services

Problem	?	Solution: Use a Cluster Accessed Using Web Services
Price a portfolio of callable bonds.	?	Price an individual bond (or a subset of all bonds) on a node, repeat as necessary.
Price a portfolio of risky bonds.	?	Price an individual bond (or a subset of all bonds) on a node, repeat as necessary.
Compute VaR and CVaR.	?	Run a set of Monte Carlo interest rate simulations on each node; collect, aggregate and display results on desktop.
Do "what-if" analysis for convertible bonds (for more details about this example, see below).	?	Construct set of considered possibilities on desktop; send subsets of possibilities to nodes; collect results, analyze on desktop.
Other.	?	Parallelize solution algorithm and develop Excel interface; use developed Web services template(s); access cluster.

For example, consider the problem of evaluating a portfolio of fairly complex instruments, say callable bonds. Assume the instruments vary in their features and hence must be evaluated independently, and the evaluation of each instrument is fairly costly, perhaps requiring about one minute on a single processor. Finally assume that beyond the portfolio value, we require portfolio greeks, delta and gamma, and approximate risk values VaR and CVaR (value-at-risk, conditional value-at-risk). This type of request arises frequently in finance. The key features are the evaluation of each instrument is independent and fairly costly, and there is little communication required.

The environment illustrated in Figure 1 can be used to solve this problem. The entries of each bond can be entered on the desktop (master) using Excel – the parameters of a bond are communicated via .NET to an idle server (processor) which computes the fair value of the instrument (and delta, gamma) and the result is communicated back to the master. On completion the master has computed the portfolio value, delta and gamma as well as short-term VaR and CVaR (using a delta-gamma approximation). Our experiments indicate that computing times can be reduced dramatically in this environment relative to a single processor. For example, the evaluation of a portfolio of 1,000 bonds required 7.5 minutes on a 64-node cluster versus almost seven hours on a single processor. We have transformed seven hours into seven minutes.

Two advantages of this approach are that a familiar portfolio manager, Excel, can be used; the preferred algorithm to evaluate a single instrument is used unchanged (repeatedly, on different processors). It is interesting to note, however, that if the instruments were simple, requiring little computation per instrument, then this approach must be modified to be effective – sending a single instrument to be priced on a processor is not effective since then there is relatively too much communication relative to the computation demands. A solution is to "bunch up" the requests to evaluate instruments. ■

Acknowledgements

The observations in this article rest on research and development performed by our group at the Cornell-Waterloo Solutions Lab in Manhattan. Especially, we thank Shirish Chinchalkar for all his help.

Appendix: A Web Service

A Web service is an element of functionality, such as a method or a function call, which is exposed through a Web interface. Any client on the internet can use this functionality by sending a text message encoded in XML to a server, which hosts this functionality. The server sends the response back to the client through another XML message. For example, a Web service could compute the price of an option given the strike, the stock price, volatility and interest rate. Any application over the internet could invoke this Web service whenever it needs the price of such an option.

1. There are several advantages in using Web services to perform computations: XML and HTTP are industry standards. This means that we can write a Web service in Java or Linux and invoke it from a Windows application written in C# and vice versa.

2. Using Microsoft's NET technology, we can invoke Web services from office applications such as Microsoft Excel. This feature is especially useful in the financial industry, since a lot of end-user data is stored in Excel spreadsheets.

3. No special-purpose hardware is required for running Web services. Even different types of computers in different locations can be used together as a Web services cluster.

4. Since the Web service resides only on the Web server(s), the client software does not need to be updated every time the Web service is modified. (However, if the interface changes, the client will need to be updated.)

5. The Web service code never leaves the server, so proprietary code can be protected.

About the Author

Dr. Thomas Coleman is chairman of the Faculty of Mathematics at the University of Waterloo (Ontario, Canada).

FOR MORE INFORMATION:

Roger Lang
Director, Corporate Relations
Cornell Operations Research Manhattan
55 Broad St.
New York, NY 10004
212-363-2915 x 13
RLang@tc.cornell.edu
www.orie.cornell.edu/manhattan