DISTRIBUTED TECHNOLOGY TECHNIQUES FOR SOLVING DYNAMIC MODELS

PAUL TURTON AND JAN M. HERBERT

Abstract. Solving large economic models requires large amounts of computational effort, as the complexity of these models increases the computational effort required in their solution increases dramatically. To examine the nature of these solutions researchers need to repeatedly solve models using different parameter sets, this compounds the need for computational effort. This paper examines the use of distributed computing as a way of providing large amounts of computational effort. It examines distributed computing projects such as "SETI@Home" which uses millions of computers supplied by volunteers to process recorded radio telescope data, the Distributed.NET project that deciphered the 64-bit RC5-64 cipher in 2002 and the BOINC project that allows volunteers to specify the projects that their PC time can be used in. The paper proposes a technique that will use distributed computing to solve dynamic models. A sample model is presented. The model can be solved using a shooting algorithm which requires a search over many candidate solutions. In the distributed approach, a central server will use a database to log potential search areas and pass these on to the distributed computers who will then run algorithms to search over candidate solutions. Once complete the results of the search will be reported back to the server.

1. Introduction

Computers have helped to revolutionise the way economists and scientists look at the world, they have allowed the production of ever more complex mathematical models that model the physical universe. These models range from the modeling of molecules and their behavior to the modeling of the Earth's complete weather systems. They can be used to predict the future, prove or disprove theories and all without the need to resort to the more traditional laboratory techniques.

Up until the 1990's if you wanted to do any work on complex mathematical models at the end of the day you would have to access some high powered computer and the more complex your model the higher the power of the computer needed to solve it. In the 1990's, because of Moore's Law [10] which states, that computing power doubles every 18 months, computing power had become cheaper and more available and as a result desktop personal computers had the power of supercomputers of a few years previously. Along with this extra available processing power the improved communication facilities offered to computer network designers by the early wide area networks protocols allowing the Internet to begin it's massive rise, has allowed these powerful computers to be connected together to share information. The combination of processing power and much improved communication has also allowed researchers to connect computers together via these networks to provide

Date: April, 2004.
Key words and phrases. Distributed Computing, Dynamic Models.

computational capabilities equivalent to supercomputer facilities for use in research projects.

In 1996 the idea of using these fast, connected, cheap computers in major research projects occurred to many people independently at about the same time, but all the projects had many similarities. They all shared the fact that they were based on client server architecture i.e. clients got some task from a server, the server handled the data distribution and central management jobs. The task was delivered across the communication media from the server to the client, the client worked on the task and returned the result to the server at a later time. The server updated a database of completed tasks and kept track of the tasks that did not return.

2. Distributed computing projects - a brief history

Distributed computing has been around for a long time in the scientific world, scientists by nature have often collaborated on projects and as such have been keen to use computers to help them in these collaboration tasks. However the use of distributed computing involving the Internet and the general public is a relatively recent event. It first appeared in the early 1990's as a solution to a lack of computing power available to researchers. These days this form of distributed computing is showing that it can provide significantly more CPU power than existing Super Computers [2].

How then has this distributed Computing evolved into the position it is in today? One problem that has been of interest to Mathematicians for hundreds of years is the factoring of large integers. To do this as quickly as possible two things are needed: a large amounts of processing power and an efficient algorithm to factor the numbers. In 1988 Lenstra, Lenstra Jr.,Manasse and Pollard used their new algorithm [1] to reduce substantially the amount of time taken to factor large numbers of the form $r^e \pm s$. To help speed the process of factoring further, the software based on their algorithm distributed the factoring tasks over a number of workstations within the DEC laboratory where they worked at the time. In order for the project to expand they needed access to more computing power and they calculated that they could *"get the equivalent performance from 300 workstations or 1200 PCs or a single high speed factoring computer"* [9]. Obviously if they could get the processing power for free then why bother with buying expensive hardware? With this in mind they extended the project to include computers outside the laboratory and as this was in the time before the widespread use of the Internet they communicated with the computer users via email. In 1990 they had about 100 CPU time donators and they were working together to factor numbers with 100 digits. This project led on to the Number field sieve project where the number of collaborating computers grew to several hundred and they were used to factor numbers in the 100+ digit range. As the number of contributing computers grew the problems associated with work distribution grew and the distribution of work via email, whilst fine for small numbers of distributed computers, became a major problem. They solved this by using a server computer to handle the distribution and collection of work and so this project became the first distributed computing project to make use of free computer time.

Another project along similar lines to the factoring of large numbers project is the search for very large prime numbers of the form $2^n - 1$, these numbers are called Mersenne primes after the French monk Marin Mersenne. In his paper

Cogitata Physica-Mathematica (1644) Mersenne stated that the numbers $2^n - 1$ were prime for $n = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127$ and 257 He had tested up to n = 19 but the numbers above this were not tested in his lifetime. It was over 100 years later when Euler verified the next number on the list, $2^{31} - 1$, was prime. Another hundred years later in 1876, Lucas verified $2^{127} - 1$ was prime. Seven years later still, Pervouchine showed $2^{61} - 1$ was prime, but this was not on Mersenne's original list. In the early 1900's Powers showed that Mersenne had also missed the primes $2^{89} - 1$ and $2^{107} - 1$. And finally it was in 1947 that the Mersenne's list of numbers where $n \leq 258$, had been completely checked and it was determined that the correct list is:

$n = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107$ and 127.

The number $2^{257} - 1$ was not prime and therefore not on the list after all. Work continued on the search for other larger Mersenne primes using more and more powerful computers, right up until 1996 the search for these numbers was restricted to the most powerful computers of the time. On 3rd September 1996 the number $2^{1257787} - 1$ (known as M1257787) was the last Mersenne prime to be found using a supercomputer, it was found by Slowinski and Gage [11] using a Cray T94 supercomputer. This was not the end of the Mersenne prime hunt though, merely a change in the type of tools used in the hunt. It was, however, the end of the use of supercomputers in the search, as the last six Mersenne primes have been found by distributed computing. The Great Internet Mersenne Prime Search project (GIMPS) has been leading the hunt for these primes. It was set up by George Woltman and was started in 1996 to continue the search for this type of prime. In November that year the 35th Mersenne prime number was discovered. The project continues to this day and the latest number to be found was on 15th May 2004, if confirmed the number will be the 41st Mersenne prime. If you would like to help with a donation of some of your unused CPU time you can find them at http://www.mersenne.org/prime.htm.

Why go to all this effort to find these prime numbers? Mersenne primes are used in cryptography to generate the keys used in the encryption process, the ability to find these keys faster means that the codes produced by the encryption process are not so secure. At the moment there are two competing encryption standards for public key encryption; the RSA (Rivest, Shamir, Adelman) standard and the Fast Elliptic Curve Encryption (FECE) which is based upon Mersenne primes and was developed by Richard Crandall [8]. The RSA standard has been around for over 20 years, but FECE although much more recent, has already gained the recognition of standard organizations such as IEEE, ANSI and ISO. The recent progress in the factorisation algorithms and the success of projects like GIMPS has forced a significant increase in the key size for the RSA algorithm. The large key size means that software implementations of RSA are slow and are not good in certain applications. Hardware implementations of the RSA algorithms require large areas of silicon to impliment and are therefore expensive and not very well suited for applications such as smart cards which rely on the fact that they are small and cheap.

The FECE algoritm is an emerging class of cryptosystems that offers more security per key bit than any other known public key scheme. With FECE the same security is obtained using an almost ten times smaller key than for RSA. This leads to fast software implementations and fast and area-efficient hardware implementations. Additionally, FECE is not a single algorithm but rather a family of algorithms. The user can choose a particular transformation from this family as a tradeoff between the speed and the security of the cipher.

The GIMPS project attracted about 4000 volunteers contributing their unused CPU time to the project, however this large number was dwarfed by the RSA code breakers project. In the late 1990's RSA, in a bid to test it's own products issued a number of challenges based on a direct attack on encrypted text using various types of encryption. In one challenge a standard cipher used by the U.S government in the 1970's was used to code a message. The key needed to unlock the message was a 56-bit binary number. There are various ways to tackle this problem but the brute force method to unlocking the message is to try all $2^{56}(7\text{x}10^{16})$ different binary numbers. In June 1997 a public distributed computing project called DESCHALL managed by Verser, Curtin and Dolske [14] hit on the correct key after only trying 20% of the available possibilities, they were rewarded by RSA a prize of $10,000. Later that year another RSA challenge which also employed a 56-bit key but using the RC5 encryption algorithm was deciphered. This time almost 50% of the key space was searched before the correct key was found. The key was found by the Bovine project organized by Beberg, Lawson and McNett [3]. This project made history by enthusing the general public in a way never seen in distributed computing up until that point. The massive response could be something to do with the fact that peoples imaginations were fired because they were using their computers to try to break Government codes, league tables of key statistics were also used to encourage the competitive spirit of contributors. The project attracted 4000 teams who, between them, contributed 26,000 pentium computer equivalents, all this computing effort was processing 7 billion keys per second at the height of the project. RSA still has contests running and the latest number to be factored was RSA-576 and it was factored in December last year. The current number being worked on is RSA-640.

In 1996 another project making use of the Internet and unused computing time was proposed. The Search for Extra Terrestrial Intelligence at home or SETI@home project was launched, like the previous projects it was also designed to make use of the unused capacity on ordinary desktop PCs connected via the Internet, it also built on the successes of the RSA code cracking project in that it employed techniques to encourage participation by the general public. In general the SETI project was set up to analyse data recorded from the Arecibo radio telescope looking for signs of intelligent life. In their paper "A new major SETI project based on Project Serendip data and 100,000 personal computers" Sullivan, Werthimer et al [13] proposed the use of "massively parallel computation on desktop computers scattered around the world". They were aware that to be able to attract the large numbers of people they needed for their project to be successful they needed to make the project attractive to the general public. In the words of the authors the project would;

(1) be the first time ordinary computer users will be able to participate in a major scientific experiment.
(2) be the largest distributed computation ever undertaken.

(3) provide for each participant the slight but captivating possibility that his or her computer may be instrumental in the detection of another technical civilization in the Milky Way.

The project is similar to GIMPS and the RSA code cracking projects in that it relied on interested users donating their free CPU time to the project. To make the client software more attractive it makes use of the fact that most computer users have some sort of screen saver running on their computer to preserve the life of the CRT displays. So the client software, as well as a performing the data analysis task required by the project also performed the relatively mundane but attractive task of screen saver. All participants in the project had to do was download the SETI client, load it onto their machines, configure it and leave it. The SETI client detects when the computer is not being used and runs its screen saver and data analysis, the program processes recorded radio telescope data from project Serendip IV. The data is a continuous tape-recording of a 2MHz signal centered on 21cm HI line. This data is broken into small chunks of 50 seconds of 20KHz bandwidth. These chunks are stored on a server and given out to client computers over the Internet to process. The project expected 100,000 computer clients to download the chunks, this would give them the computing power of a fraction of a supercomputer. In all the recording amounts to 25GBytes of data to process every 11 hours. It was expected that each chunk of data would be processed in approximately 'several days' after which the client computer would report the results. In 1997, when the project was conceived, it was envisaged that with 180,000 users they could process 43% of the data recorded. The project actually attracted millions of participants and is still running today, but under the banner of a more general project called Berkeley Open Infrastructure for Network Computing or BOINC for short. To date SETI@home has used 1.5 million years of CPU time [2] and has shown just how successful Public Computing can be. They have, however, not found any evidence of extra terrestrials!

It is still difficult for researchers to make use of this distributed computing resource, one way this problem may be solved has been outlined by the BOINC project. In his paper "Public Computing: Reconnecting People to Science" [2], Anderson outlines the attractions to scientist and researchers of tapping into the massive amount of CPU time out there on the Internet. He also points out the social and technical aspects of public computing. Socially public computing is only successful if people participate, SETI@home attracted 4.6 million participants and as such can be described as being very successful. The technical aspects of public computing require "adapting an application program to various platforms, implementing server systems and databases, keeping track of user accounts and credit, dealing with redundancy and error conditions" [2]. The BOINC project is an attempt to offer researchers a tool kit of parts to enable them to convert existing projects to a public computing project and as such is worth exploring if you are hoping to make use of public distributed computing.

To sum up: to convert a computing problem into a public distributed computing successfully the problem needs to be broken down so that –

(1) the processing is not sequential
(2) it does not require large amounts of bandwidth to transmit it over the net.

3. Solution of dynamic models

The distributed computing technique has been very successful at solving a number of scientific problems. In the rest of this paper we consider it for solving dynamic economic models. We use a particular continuous-time differential equation model.

As we have seen for a problem to be successfully converted into a distributed computing problem it must not be sequential and it must not require large amounts of bandwidth to transmit over a network. The model we are interested in described in Section 4 falls into this category. The model has a stable state and we are interested in how the model moves from one stable state to another when the model is exogenously shocked. The problem is to find the transient dynamics of the model to the new steady state. The model has 'jumping' variables which jump the model onto the stable manifold and the transient dynamics are on this manifold. We know the initial conditions of the non-jump variables, and the terminal conditions of the jumping variables. To find the solution trajectory we use a shooting algorithm i.e. we turn the two-point boundary value problem into a initial value problem. The shooting algorithm works by searching or taking a 'guess' at the initial conditions of jumping variables. Once these conditions have been found we have the trajectory of the stable solution of the model. If the initial condition values we guessed do not give a trajectory to the stable steady state then the guess was wrong and we must try again. As you can imagine the amount of guessing or searching necessary to solve this model will vary but it could be very large. The biggest part of solving this problem is the searching and as can be seen this can easily be parallelised. The bandwidth needed to transmit the problem over the network is small, the initial model, containing search algorithm and differential equation solver can be transmitted over a network in no more the a few hundred kilobytes. The results of searches and computations will be small and possibly not more than a few kilobytes.

Many computers searching over different areas reporting their success or otherwise to a central server is a very good use of the distributed computing model discussed earlier. As such using the this model for the solution of dynamic models looks to be ideal. The next section of this paper examines the dynamic model that we are interested in examining.

4. The Dynamic Model

The dynamic model we use as a test model is that presented in [7]. The model concerns the investment decision of a profit-maximizing firm with n types of capital. The firm faces a Cobb-Douglas production technology, and has adjustment costs associated with the installation of new capital. The magnitude of these adjustment costs is governed by the magnitude of parameters, b_i. The decision of the firm can then be summarized as follows: Choose the I_i's to maximize:

$$V = \int_0^\infty e^{-rt}[F(K_1, K_2, \cdots K_n) - \sum_{i=1}^n I_i]dt \tag{4.1}$$

subject to:

$$\dot{K_i} = I_i - b_i(\tfrac{I_i^2}{K_i}) \tag{4.2}$$

$$F(K_1, K_2, \cdots K_n) = a \textstyle\prod_{i=1}^n K_i^{\alpha_i} \tag{4.3}$$

with initial capital stocks $K_i = K_{i,0}$ for the $i = 1, 2, \cdots, n$ capital stocks, and where:

$\sum_{i=1}^{n} \alpha_i < 1$;
K_i represents the real stock of capital of type i;
I_i represents the real level of investment of type i;
$F(K_1, K_2, \cdots K_n)$ represents real output;
r represents real interest rate (assumed exogenous); and
a, b_i and α_i represent exogenous parameters.

The dynamics of capital accumulation reduces to:

$$\dot{q_i} = [r - b_i(\Lambda(b_i, q_i))^2]q_i - F_i, \qquad \text{for } i = 1, 2, , \cdots, n \tag{4.4}$$

$$\dot{K_i} = \Lambda(b_i, q_i)[1 - b_i\Lambda(b_i, q_i))]K_i, \qquad \text{for } i = 1, 2, , \cdots, n \tag{4.5}$$

where:

$$F_i = F_{K_i} \tag{4.6}$$

$$= \frac{a\alpha_i}{K_i} \prod_{i=1}^{n} K_i^{\alpha_i} \tag{4.7}$$

$$\Lambda(b_i, q_i) = \frac{q_i - 1}{2b_i q_i} \tag{4.8}$$

Note that the initial conditions for the capital stocks (K_i's) are known, but that the initial conditions for the co-state variable (q_i's) are not. These variables must 'jump', when the model is shocked, so that the model's solution trajectory is on the stable manifold for the solution. For more details see [6, 12, 5, 7].

With different numbers of capital stock and different values of the parameters, it is possible to generate many different parameterised models. The models can be divided into classes of interest depending upon such features as dimensionality (parameter n) and degree of nonlinearity (the b_i's and α_i's).

Solving each model for an exogenous shock to interest rates (r) is a two-point boundary value problem that requires the solution to many possible candidates to the dynamic model given by equations 4.4 and 4.5. Each candidate solution will have a different 'guess' for the initial conditions of the co-state variables.

The solving of each candidate model is computational intensive and requires an algorithm. Most algorithm's require a numeric search as well as a numeric differential equation solver. But once the solution is found only a small amount of information is required to replicate the solution trajectory. This information is the initial conditions for the co-states (q_i's) and together with the initial conditions for the capital stocks (K_i's) they define the model as an initial value problem that can be easily solved with a single pass of a numeric differential equation solver.

Thus the model is an ideal test model for a distributed computing technologies as low bandwidth is needed for the results, but the calculation of the results requires substantial computing effort.

5. Setting up a distributed system to solve dynamic models

How are we going to set up a system that will allow computers connected via the Internet to donate their un-used CPU time to a project that will use distributed computing to solve dynamic models? It is the intention of this research to show that the idea of using this method of solving dynamic models is feasible and useful, as such an off-the-shelf solution would be the quickest way to do this. To this end, the first stage in the implementation of this work will be to set-up a BOINC [4] based project. On their web site, the BOINC team setout the requirements for creating the server platform for a distributed computing project using volunteer computer resources. An overview of those requirements is presented here.

The operating system (OS) required for the central server is Unix or Linux, many versions of these OS are supported. Once the correct OS has been set up on the server a web server that will allow the project web site to be hosted on the this machine must be configured, this is the visible part of the project that will be seen by project contributors on the Internet. Any web server software can be used but the most likely will be the Apache web server which comes free with most versions of Linux. The BOINC software supplied, needs to be compiled and configured, this requires the installation of the correct compiler (please see http://boinc.berkeley.edu/software.php),again, with most versions of Linux this compiler is free. The project database is based on MySQL database software (http://www.MySQL.com) which can again be obtained free under the GNU general public licence agreement. This software must also be installed on the project server. When the database up and running the next thing to do is to set up the application, in our case this will need to be written, this will be the search algorithm, differential equation solver and communication with the server software. The application can be written in any language but the BOINC web site gives examples of how to include FORTRAN applications with the native C++ language that the sample project is written in. As our application software is not available at the moment we will test the software set up using the sample application supplied with the BOINC software.

With the Web site up and the application software available it just remains to create work units for the client computers to process, in our case this will involve defining the parameter-sets and search areas for the client computers to work on, these are stored in the project database using the tools provided. With the work units defined the setup of the basic project server is complete all that remains to to create the clients, initially this can be done on the server computer, but ultimately the clients will obviously be on different computers. The clients will need to setup accounts on the project server and be assigned work by the server. Once configured the client will process the test files and credit will build up on the server showing that the clients are processing work.

The time scale involved in setting up and testing the above server/client systems will be of the order of a few weeks. To write, optimise and test our own application software would be approximately six months. It then remains to recruit helpful members of the public to provide those free CPU cycles we need to show that Distributed Technology Techniques for Solving Dynamic Models is a useful way forward in providing more computing power for the investigation of Dynamic Model behavior.

6. Conclusion

In this paper we have presented an approach to the distributed solution to dynamic economic models. The paper contains a discussion of how distributed computing has successfully been used in a number of scientific projects, for example the 1.5 million years of free CPU time have been made available to the SETI project.

We have put forward a sample dynamic model problem that we intend to solve using this approach. The paper gives details of the how we intend to solve the problem. If you would like to be involved in the project by denoting your free computer time then email paul@itb.edu.bn.

References

1. M. Manasse A. Lenstra, *The number field sieve*, Proc. Symposium on the Theory of Computing, Baltimore, May 1990.
2. David P. Anderson, *Public computing: Reconnecting people to science.*, Conference on Shared Knowledge and the Web, 17-19 Nov 2003, Residencia de Estudiantes, Madrid Spain.
3. Lawson McNett Beberg, Web Site – http://distributed.net/, October 1997.
4. David Anderson Karl Chen Seth Cooper et Al, *Berkeley open infrastructure for network computing (boinc)*, Web Site – http://boinc.berkeley.edu/, 2004.
5. Ric D. Herbert, *Exploiting model structure to solve the dynamics of a macro model*, Computational Economics **21** (2003), no. any, 203–207.
6. Ric D. Herbert and Peter J. Stemp, *Solving the dynamics of a non-linear representative agent model*, Mathematics and Computers in Simulation **59** (2002), no. 1/3, 95–104.
7. Ric D. Herbert, Peter J. Stemp, and William E. Griffiths, *Assessing two common approaches for solving models with saddle-path instabilities*, Mathematics and Computers in Simulation (2004), in press, acceptance date of Feb 4, 2004.
8. R.E. Crandall NeXT Computer Inc, *Method and apparatus for public key exchange in a cryptographic system.*, Sep. 17 1991, U.S. Patent 5,159,632.
9. Arjen K. Lenstra and Mark Manasse, *Factoring by electronic mail.*, In Advances in Cryptology ("EUROCRYPT 1989"), 1990, Lecture Notes in Computer Science. Springer-Verlag.
10. Gordon A. Moore, *Cramming more components onto integrated circuits*, Electronics (April 1965), Vol 38, No.8.
11. Slowinski and Gage, Web Site – http://www.utm.edu/research/primes/notes/1257787.html, September 1996.
12. Ric D. Herbert Peter J. Stemp, *Calculating short-run adjustments: Sensitivity to non-linearities in a representative agent framework*, Journal of Economic Dynamics and Control **27** (2003), no. 3, 357–379.
13. Wertimer et al Sullivan, *A new major seti project based on project serendip data and 100,000 personal computers*, April 1997, Proc. of the fifth Intnl. Conf. on Bioastronomy IAU Colloq. No.161.
14. Curtin Verser and Dolske, Web Site – http://www.interhack.net/projects/deschall/, June 1997.

Institute of Technology, Brunei
E-mail address: paul@itb.edu.bn

The University of Newcastle, Faculty of Science and Information Technology, Newcastle, NSW, 2308 Australia
E-mail address: jan.herbert@newcastle.edu.au