TOPICS IN THE DESIGN OF A PARALLEL TLM SOLVER

Sasse H. G., Duffy A.P.
De Montfort University, Leicester, UK
(Principal contact: hgs@dmu.ac.uk)

Abstract: Numerical modelling is a widespread tool in the EMC community. However, significant demands are placed on computational resources. This paper discusses the problems associated with the implementation of a parallel modelling package, particularly in respect of the availability of suitable hardware. The use of existing networks of available computers is considered as a solution to this problem and a number of requirements for such a distributed design are set out and justified.

1. Introduction

Topics of interest to modellers in EMC community include novel antennas, coupling into complex cavities, reverberant chambers, etc. To model these accurately makes significant demands on simulation time and computer memory, particularly if high accuracy is required. Many of these problems are prohibitive for modern workstations, unless the modeller is prepared to wait a long time for the results.

Although workstations are generally not sufficiently powerful for the most complex problems, most organisations have a surfeit of untapped computing power in the form of PCs, many of which are idle outside working hours. If it were possible to harness this power for simulations it would enable more accurate models to be simulated, and models to be run in less time.

This paper discusses the issues raised when trying to access this power.

Firstly this paper will provide a simple introduction to the Transmission Line Modelling (TLM) technique, which will be used as a vehicle for the discussion of the applicability of parallel solvers in the rest of the paper. Some of the difficulties presented by TLM will be discussed and then the further issues raised in implementing the algorithm in parallel will be examined. The paper then discusses key issues in parallelisation and strategies for solution.

2. The Modelling Environment

2.1 Numerical Modelling

There are a number of reasons why one would wish to model radiating systems numerically. Existing hardware may need to be modelled so that the fields produced can be studied easily. Sampling fields with a probe is time consuming and is less intuitive than the visuals a computer can produce. Systems which do not yet exist may be modelled to save on construction costs of prototypes. It is generally cheaper to alter a model than to rebuild the piece of equipment.

Modelling may be used to determine how a system will perform in a given setting, such as the placement of an antenna at different positions on a vehicle, or in determining the susceptibility of these vehicles to various environmental EM fields.

The TLM method is particularly suited to such problems as mode stirred chambers, waveguides and patch antennas. This is due to the ease with which areas of metal can be modelled in TLM, which is more complex in other techniques such as Method of Moments.

2.2 Transmission Line Matrix (TLM) Method.

TLM modelling is achieved by treating the region of interest (the workspace) as a collection of rectangular sub-regions (nodes) of sufficiently small size that the material properties of that area may be considered nearly uniform.

Each of these nodes may then be modelled as a cuboid with voltages across the faces, with these voltages representing the electromagnetic field about the node as illustrated in Figures 1 and 2.

The node then propagates these signals by means of internal transmission lines to its other faces so that a signal impinging on one face of the node gets scattered to the adjacent and opposite faces in the manner of

Huygens' model of light propagation[1], where each point is regarded as the generator of a new wavefront.

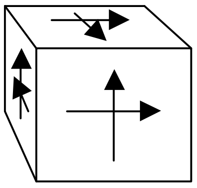

Fig. 1 A cuboidal representation of a 3D TLM node

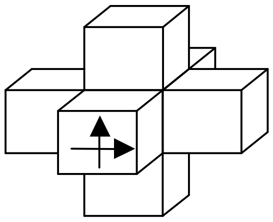

Fig. 2 A conventional representation of a 3D TLM node

Once the internal scattering is complete, the nodes then pass the voltages on to their immediate neighbours, in a "connect" phase.

The sequence "scatter, connect" is repeated as many times as necessary for the simulation in hand. Thus this technique is a time-domain method.

The internals of the node are represented as a circuit of transmission lines connected together in a number of possible different ways. These yield a "black box" model of the node, which may be held in a matrix specifying how a voltage on one port gets scattered to other ports. There has been a lot of research in this area and models that have been created and used successfully include shunt and series nodes, symmetric nodes, condensed nodes, super symmetric condensed nodes, nodes representing space with internal wires, and many others. The details of such models are beyond the scope of this paper, whose concern is the issues of implementing TLM in parallel.

2.3 Problems with TLM

The main problem with this simulation method is that the time to run accurate simulations of large problems is very long. "Days" seems to be considered normal in the TLM community, though this is no criticism of the technique itself -- many other techniques can take as long. Clearly a reduction in simulation time would allow users to make more progress and would open up other techniques where repeated trials are necessary for design optimisation. The use of genetic algorithms or simulated annealing, for example, could lead to new designs of antenna. but such techniques would take prohibitively long at present, if performed on a single machine. For example, a genetic algorithm may take hundreds of generations to reach a solution, resulting in thousands of simulations even for a small population size.

The problem is exacerbated because we are interested in sizable problems, where a problem is considered sizable if it has a lot of detail or because it encompasses a large space, or possibly both. If there is a lot of detail then the nodes must be very small, so many are needed. If the problem encompasses a large space because measurements points or significant structures that must be modelled are far from the antenna, then there is no choice but to use a large number of nodes. To some extent non-uniform gridding can help in these cases, at some cost to precision in the results.

Having identified a modelling technique and how it could be utilised for increased benefit, such implementation is limited by available computing power. Supercomputers are beyond most academic institutions and commercial entities. An alternative solution which is affordable is therefore a justifiable goal. The next section addresses one promising approach to this goal.

3. Parallel Processing

The aforegoing discussion, in which each of the nodes is described as performing the same task, at the same time, has demonstrated that TLM is inherently parallel in nature. Clearly, to have a single processor working though the computations for each node in turn is therefore a suboptimal solution. Furthermore, since we have already discussed the requirements for a speed increase. parallel processing is a logical step.

There are a number of different models of parallelism that might be considered. For example, operating systems such as Unix run many tasks in parallel, and many programming languages support threads now, however these techniques are not true parallelism because one processor is doing all the work, actually in series. Other methods involve having many processors share memory between them, and these must therefore be located on the same bus usually within one enclosure, sharing a motherboard. This involves having hardware designed specially for the purpose, the design considering issues such as the use of shared resources. Other systems use processors which are designed to communicate with their immediate neighbours over specialised channels, such as Transputers, for example.

3.1 Constraints on Parallelisation

The problem of the cost of supercomputers, mentioned above, means that specialised hardware is not an ideal solution to solve this problem Buying time on a massively parallel machine would not grant one full-time access to it, impeding the overall progress. However, the presence of underutilised machines together with an internal network and the internet, offers a practical alternative with limited capital outlay. In many cases capital outlay has already occurred.

Such an approach sets constraints on any design of a parallel modeller.

The machines on these networks will be diverse, being different types of hardware and running different operating systems or different versions of one operating system. This means that programs designed to make use of them must be particularly portable.

There will be a wide variation in performance between machines, for example between an Intel 80386 machine and an AMD Athlon.

The programs cannot rely on external libraries, because one cannot be sure that the libraries will be available, or that they are the correct version, or that subsequent versions will be compatible with the software. Thus any attempt to upgrade, or refusal to upgrade at a remote site, would cause the software to be unusable there. Attempting to install correct versions of the library may not be possible if other software already in use on the machine requires that the version be different to support some feature. There may also be conflicts between different libraries.

It cannot be assumed that machines will be well supported, and so the code must need minimum intervention at a local level. Continuing requests of the remote site administrators for maintenance work to support the software will be unwelcome.

The program, being a "guest" on the distant machine, must not make it impossible for the rightful owners of the machine to use it; the software must not overutilise CPU and memory, and it must not impede normal work. To achieve this it will be necessary for machines to provide a service intermittently, resulting in a need to pass state information from machine to machine to enable the simulation can continue seamlessly.

3.2 Consequences of these constraints

The consequences of these constrains is that we consider it inappropriate to use libraries such as MPI, & PVM[2,3].

The development of a multi-faceted program may be eased by the use of different languages. For example FORTRAN is particularly suited to mathematical computation, whilst for parsing definitions of the problem, it may be more appropriate to use yacc, lex and the C programming language. However, such an approach is not helpful in a widely distributed program because it creates too many dependencies on external software, such as compilers (which for more obscure languages may not even be present -- Solaris no longer ships with even a C compiler for example).

It is necessary to ensure that the simulation runs "out of hours", which will be different hours dependent on time-zone, and may need to be flexible. Someone may wish to allow use of their machine while they are in a meeting for example. In the ideal case the machine could detect that it is idling, and could make itself available for the distributed modeller without user intervention.

Machines of differing performance necessitates careful consideration of the way the problem is shared between machines.
Communication delays will vary between machines, because some will share a local network, and others will be separated by several routers.

Clearly in such an inhomogeneous, time variant system we have a need for load balancing and fault tolerance, and a need to minimise communication between machines.

3.3 Proposed solutions

This section addresses the issues raised in the previous section and suggests strategies for their solution.

3.3.1 Special hardware

Past work on parallel TLM has often used special hardware. A number of papers discuss using Transputers Hui, Christopoulos et al [4] have used transputers Tan & Fusco [5] have used a DAP So, Eswarappa and Hoeffer [6] have used a DECmpp 1200P and a Connection Machine. These machines have their own internal networks, and present different problems from those discussed here.

A number of papers discuss using linked machines of different types, for example [7], but such heterogeneous networks have mainly been workstations used to support special hardware.

3.3.2 Segmentation

If the problem is to be divided up among many machines there are a number of ways in which this division can be undertaken. For example Parsons *et al* [7] consider networks of workstations, and they considered that communication is 'expensive' in terms of time, because accessing information across a network is considerably slower than if the same information is available in the local machine's memory. The logical extension of that it is important to put as much of the workspace volume into the memory of each machine as possible, whilst minimising the communicating "surfaces" between the machines. They conclude that the volume:surface-area ratio must be maximised for each segment of the problem.

Pollmeier *et al* [8] discuss machines with different performances, suggesting loading could be pre-computed. In our case the performance will be time variant, so this result is less applicable to our case. Most other papers simply divide the space into slices, usually because of the processor architecture.

One particularly important aspect of segmentation is dynamic load balancing, which is discussed in section 3.3.4.

3.3.3 Networking

Given the foregoing discussion it is clear that a different approach will be necessary to satisfy the above constraints. This does not invalidate these other approaches, because they were based on different resources being available to these authors.

To ensure generality and to satisfy the described constraints it is proposing to use ethernet as the communications medium between machines, both at a local (LAN) level and at the global level where this would be the Internet.

The selection of the protocol is an important choice. Most of the internet uses TCP/IP for its traffic. This is a lossless protocol, and communication is transparent on it. Packets are guaranteed to arrive, and to arrive in the correct order. However, the communication between nodes in the TLM model works at the neighbourhood level, with each node passing information to its immediate neigbours on a peer to peer basis. TCP/IP requires that the two ends of a communication channel have a client/server relationship. This would mean that each node would need to change its role depending on whether it initiated the communication or not. This is too complicated to manage in a general sense, and also TCP/IP is a rather computationally expensive protocol, compared to the User Datagram Protocol.

The User Datagram Protocol (UDP) is not a client/server protocol, but it has the disadvantages that it is lossy; packets may arrive incomplete or not at all, and furthermore they may arrive in the incorrect order. This means that structure must be imposed on the packets to correct this when it happens.

Protocol design is a large topic in itself, so in the first instance a simple, or even simplistic, ARQ scheme (automatic request for retransmission) is being considered. Packets are numbered so that sequencing may be checked, and the data in the packet is given a 16-bit checksum. Clearly a more efficient protocol may well be possible.

3.3.4 Load Balancing

An important problem in any parallel system is load balancing.
The workload must be spread evenly among the machines, where "evenly" means that the machines will finish at the same time, so none are kept waiting. If the problem space can only be divided in 2 this is fairly easy. The boundary will move to make the areas proportional to the performance of the processors. Where there are four machines the situation can be more complicated, because the boundary between any two processors could be moved independently of all the other such boundaries, e.g. Figure 3.

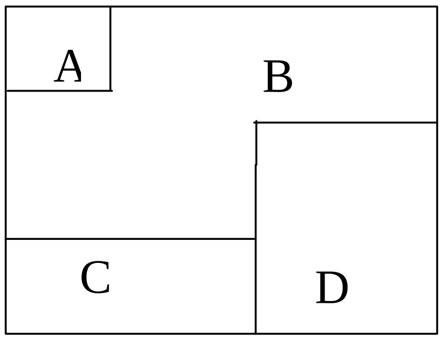

Fig. 3 A potential interprocessor boundary problem. A, B, C, D represent individual processors

If things are kept to the simplest case, where there are only orthogonal planar boundaries dividing up the space, then some useful analysis can be performed, for example consider the case of four processors, as in Figure 4.

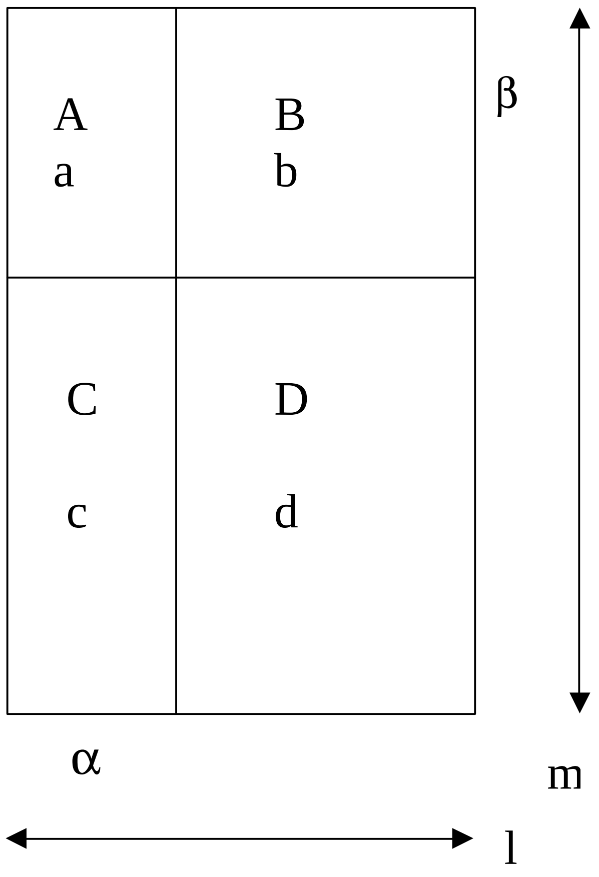

Fig 4 Four processors with planar boundaries

If scattering, and connecting within one processor is considered as one cost, and the connections to external processors as another cost

$$\text{Cost} = a \cdot a \cdot b + (a+b) \cdot b + b \cdot (l-a) \cdot b) + c \cdot a \cdot (m-b) + (c+d) \cdot (m-b) + b * (l-a) \cdot (m-b)$$

So

$$\P\, cost / \P\, a = a \,.\, b + (a + c) - b \,.\, b - (b + d) + c \,.\, (m - b) - d \,.\, (m - b)$$

showing that as alpha increases the load on *A* and *C* increases. The situation is symmetrical.

Clearly these examples don't take account of the geometry being modelled. For example bulk volumes of metal within the workspace will reduce the load on processors, because the electromagnetic waves do not propagate through the metal, so there is no need to perform scatter and connect operations within a region of bulk metal.

Also, the above discussion assumes that the boundaries are kept as planes. The more interesting case, where a region is non-rectangular, introduces more problems. However, this is not discussed here. If boundaries are moved in this case would could end up with a very fragmented space where the cost of communication may be greater than that of processing.

Such unusual arrangements may result if it proves simpler to process one type of dielectric region on one machine, with a different dielectric region adjacent also being modelled. They may also occur where a change in loading on machines results in parts of the boundaries being moved, unless the movement of boundaries is handled with some sophistication. So the segmentation of the space is non-trivial, and may be amenable to AI or optimisation techniques. We believe this is an area which could benefit from more research

3.3.5 Characterising Performance.

The foregoing discussion on load balancing assumes that processor performance is known. For this to be the case a measure of performance must be available. It is clear that CPU power effects both calculation and interprocessor communications, because of the computational cost of whatever protocol is in use at the time. Also the scatter is always followed by connect, so that this forms a critical path in the TLM process.

These two costs should be lumped together as a first approach, so that a signal goes into a processor, is processed, and a different signal comes out. The performance of the processor is a function of this "trip time".

This distributed computational model with signals taking trips between different processors is somewhat analagous to the foraging that takes place in ant colonies. In an ant colony worker ants leave the nest in search of food, leaving a pheromone trail to record their path, which they and other ants may follow. If the source of food is particularly good, then when the ant returns to the nest it communicates this to other ants, so many set out along the trail, each ant strengthening the trail. The pheromones evaporate with time, so that routes to old, depleted sites disappear. If processor performance is modelled as the quality of a food source, this seems to suggest a suitable may of load balancing a distributed system. Indeed, Schonderwoerd [9] has used them for load balancing.

3.3.6 A Proposed Solution

Considering the above constraints in turn, the authors propose the following system.

The need for diverse platforms means that a language is needed that is very portable. Because this is an experimental setup, and we wish to make changes easily, the system is currently being developed in Ruby, a fully Object Oriented language created by Yukihiro Matsumoto [10]. This language has a good selection of standard libraries, making dependence on other libraries largely unnecessary. It also works on Unix, Linux, and the various varieties of Windows. Although the language does not compile to native code, it still performs very well.

The Object Oriented nature of Ruby has meant that testing has been simplified, and that design changes have been easier to manage because of encapsulation. It has good support for networking, and the UDP protocol in particular. Data is passed using the Marshal module to convert the data into strings for transmission, and these are broken into numbered packets, handled by a simple ARQ scheme [11]. It will be possible to experiment with data compression of the strings, to improve the performance of the solver.

Functions related to time, such as determining when it is "out of hours" and for determining the relative performance of machines are well supported in the language, easing the development of time-variant behaviour.

4. Discussion

Success in producing a parallel TLM will mean that larger spaces could be modelled, and work on more detailed structures such as the effects of radiation on the human head, or on more realistic models of vehicles would be achievable.

One of the problems faced by modellers at the moment is that to import a model from a CAD system into an TLM modeller a lot of simplification must be done. This has to be done by hand, and on the basis of knowing which details are important. If all the details could be kept, this long and tedious step could be avoided.
Also, search based design could be applied to more problems. Such techniques as Genetic Algorithms are impractical when a single simulation of one model takes days, and populations in a genetic algorithm may run into the hundreds, and the number of generations needed may be thousands to achieve a stable, effective design. Parallelism in the genetic algorithm itself can help, in that all the individuals in a population could be modelled at the same time.

Although the discussion has laid out some guiding principles for a TLM solver, this set of principles has equal standing when considering other modelling methods such as Method of Moments, Finite Difference time domain, Finite Element Analysis, etc.

It is clear that there are still a number of questions open in this field, such as:

- What are suitable protocols for use in distributed systems?
- What security measures against deliberate attack on a parallel system can be achieved without sacrificing performance?
- How can division of the workspace make intelligent use of the characteristics of the problem being modelled? Are artificial intelligence techniques useful in this application?

5. Conclusions

This paper has described TLM and its parallel nature and some of the problems of implementation of a parallel simulator, in particular load balancing, communications and reliability. Solutions have been suggested to these problems based on the use of the internet to create a more powerful system. It has been shown that there is scope for further research in this area, particularly in communications protocol design, load balancing and segmentation with consideration given to the properties of the materials being modelled.

An approach using the Ruby programming language, based on the UDP protocol using a simple ARQ scheme is currently being investigated by the authors.

From the observations discussed in this paper it is clear that there is still much work to be done in the field of designing a parallel TLM solver. However, a number of directions have been proposed and the potential benefits of achieving a flexible parallel solution are not inconsiderable.

6. References

[1] Huygens "Traite de la lumiere" , Lieden 1690

[2] Pacheco, P S., Parallel Programming with MPI, Morgan Kaufmann Publishers, Inc. 1997

[3] Sunderam VS PVM: A Framework for Parallel Distributed Computing Concurrency: Practice and Experience December 1990. Vol 2 No 4, pp 315-339.

[4] Fung KK, Hui SYR, Christopoulos C, Concurrent programming and simulation of decoupled power electronic circuits IEE Proceedings - Science Measurement and Technology 1996 Vol 49 pages 1-13

[5] Tan CC, Fusco VF TLM Modelling using an SIMD computer International Journal of Numerical Modelling - Electronic Networks Devices and Fields1993 Vol 6 pages 229-304

[6] So PPM, Eswarappa C, Hoeffer, WJR Parallel and Distributed TLM Computation with Signal Processing for Electromagnetic Field Modelling International Journal of Numerical Modelling – Electronic Devices Networks and Fields 1995 Vol 8 Pages 169-185

[7] Parsons PJ, Jacques SR, Pulko SH, Rahbi FA TLM modelling using distributed computing IEEE Microwave and Guided Wave Letters 1996 Vol 6 page 141-142

[8] Pollmeier K, Burrows CR, Edge KA Partitioned Simulation of Fluid Power Systems - An approach for reduced Communication between processors Proceedings of the Institute of Mechanical Engineers Part 1 - Journal of Systems and Control Engineering 1996 Vol 210 No 14 pages 221-230

[9] Schonderwoerd R, Holland O, Bruten J Ants for Load Balancing in Telecommunication Networks HP Labs Technical Report HPL-96-35 1996

[10] Thomas D, Hunt A, Programming Ruby - The Pragmatic Programmer's Guide, Addison Wesley, 2001, ISBN 0-201-71089-7

[11] Stevens, W. Richard, UNIX network programming, Prentice-Hall International, 1990