PARALLEL SPATIAL INTERACTION MODELLING ON THE KSR1 - 64 SUPERCOMPUTER

Stan Openshaw and Robert Sumner

WORKING PAPER 95/15

SCHOOL OF GEOGRAPHY • UNIVERSITY OF LEEDS

Parallel spatial interaction modelling on the KSR1-64 supercomputer

Stan Openshaw, School of Geography, University of Leeds, Leeds
Robert Sumner, Silicon Graphics UK, 1530 Arlington Business Park, Theale, Reading

Summary

The paper demonstrates some of the benefits that high performance computing has to offer geographers. It reports the results of running a very large spatial interaction model on 1991 journey to work data for Britain on a 48 processor parallel supercomputer compared with a UNIX workstation.

Introduction

The entropy-maximising spatial interaction model is widely used in a range of application areas. It is applicable to the modelling of many different types of flow data in both research and applied contexts; The purpose of this paper is to describe various parallel implementations of a basic model as an illustration of some of the benefits that high performance computing (HPC) can offer computationally minded geographers.

The idea of developing a parallel spatial interaction model is not new. Harris (1986), discussed how (in theory) it could be ported onto a parallel machine. Openshaw (1987) presented some further ideas regarding vector supercomputers. However, it seems that not until the early 1990s were there the first large scale parallel implementations of the spatial interaction model developed, see George (1993). The reasons for using parallel hardware is to allow much bigger datasets to be processed more speedily than previously and with fewer simplifying assumptions (Birkin et al, 1995). In the spatial interaction model the compute load is an approximate power function of the number of zones, small numbers of zones run well enough on a PC but large numbers (i.e. a few thousand) require high performance computing hardware in the form of vector or parallel supercomputers. This is now important because the availability of flow data has rapidly increased in terms of both size and frequency following the GIS revolution of the mid 1980s. Traditionally flow data was hard to code but, modern methods of spatial data management and particularly the automatic coding of origin and destination addresses via post-codes has created the prospect of UK flow data tables with up to 1.6 million origins and destinations becoming available. However, currently the biggest public domain flow matrices are those from the special workplace statistics (SWS) of the 1991 census. It is now possible to obtain a 10,764 by 10,764 matrix of journey to work flows between all wards in Britain and these data can be disaggregated in various ways. This paper presents the results of running a parallel spatial interaction model on these census data.

With all spatial models, there is a general need to be able to use smaller and smaller zones in order to improve the levels of geographical resolution that are possible. Hence spatial interaction models that are based on the 10,764 wards might be expected to provide a much finer and improved level of spatial representation than would a model based on 468 districts or 64 counties. The substantive justification for this modelling exercise with the 1991 ward based journey to work data was to use the model to scale up to 10% journey to work sample data to 100%; thereby removing effects of sample zeros, to create a simulated zero unemployment journey to work dataset that removes the systematic bias caused by regional differences in the levels of unemployment in Britain and to examine the predictability of changes in employment and population distribution in the UK. These results are reported elsewhere. Other more computer science related reasons for parallelising the spatial interaction model include: interest at the levels of performance and scalability that can be obtained, as a challenge that can be used to test different parallelisation strategies; as a means of bench marking the performance of different parallel architecture's; as a source of inspiration leading to the development of new models, and more generally as a basis for stimulating new areas of research involving the computer simulation of complex human systems.

Description of the models and data

Attention was focused on two different types of spatial interaction model: (1) singly and doubly constrained versions of the standard entropy model and (2) a two parameter variant.

The basic doubly constrained model with a negative exponential deterrence function is written as:

$$T_{ij} = A_i B_j O_i D_j \exp(-b c_{ij})$$

$$A_i = (\sum_j B_j D_j \exp(-b c_{ij}))^{-1}$$

and

$$B_j = (\sum_i A_i O_i \exp(-b c_{ij}))^{-1}$$

where

T_{ij} (trips) is the number of trips between origin i and destination j:

O_i (origin) is the number of trips leaving origin i;

D_j (destination) is the number of trips arriving in destination j;

c_{ij} is the distance (cost) from i to j.

b is a parameter to be estimated.

The A_i and B_j are balancing factors designed to ensure that the model is constrained at both ends simultaneously. A singly constrained version is produced by leaving out either the A_i or B_j terms. A two parameter version was produced by changing the deterrence function to:

$$\exp\left(-\left(b_1 c_{ij} \delta_{ij}^1 + b_2 c_{ij} \delta_{ij}^2\right)\right)$$

where:

$$\delta_{ij}^1 = 1 \; if \; i = j \; else \; 0$$
$$\delta_{ij}^2 = 1 \; if \; i \neq j \; else \; 0$$

This gives the intra-zonal trips a different parameter value, $ß_1$, compared with the inter-zonal trips value of $ß_2$. The justification is that spatial interaction models are designed to model the inter-zonal flows for which the c_{ij} values makes sense. However, they are also widely expected to model intra-zonal trips for which the c_{ij} are at best suspect (viz. what is a sensible intra-zonal distance value to use?). One solution is to delete all intra-zonal trips but this has a global impact on the model since the attractiveness measures of the origins and destinations are changed. Hence the use here of the two different ß values.

The data

The 1991 SWS journey to work data for the 10,764 wards of Britain is used as the dataset for the study. The trip matrix of 10,764 by 10,764 is highly sparse as only 0.05% of the cells have non-zero values. The sparseness is a natural feature of the journey to work data; for example not many people who live in wards in Glasgow and will work in Plymouth. Some of the zeros are real but an unknown number are sample zeros and reflect the 10% sample nature of the data. The distance data is computed from ward centroids as straight line distances and for present purposes the intra-zonal distances were arbitrarily set to one half the distance of the nearest ward.

Serial code

Conventional Fortran code was produced for the two models and then modified to cope with the 10, 764 origin and destination zones of the 1991 ward level journey to work data. The three matrices used in the model to hold trips, costs, and the deterrence function could not be stored in the memory of a standard UNIX workstation, since they would require about 1.5 GByte. A solution to this is to store the trip matrix as a singly linked list which reduces the storage needed to about 5 Mbyte and to recalculate the values of the cost and the deterrence function matrix every time they are needed. Unlike the trip data, the cost and deterrence function matrices are dense and cannot be stored in sparse format. Recalculating the cost and deterrence function values dramatically reduces the memory requirements, however, this is achieved at the expense of a large increase in the amount of theoretically redundant computation . On a Sunsparc 10/41 workstation the singly constrained one parameter model took 17.66 hours to run. The non linear optimisation routine performed 29 model evaluations in the search for an optimal parameter. The doubly constrained version ran for 264.17 hours. It is clearly difficult to make much use of a model that runs for 11 says before a single result is obtained. The question now is how much faster would a parallel version run.

Porting the spatial interaction model on to the KSRI parallel computer

The Kendall Square Research KSR1 parallel super computer at Manchester University is a virtual shared memory multiprocessor with 64 processing units (cells). It is an MIMD architecture (Braunl, 1993) with a strongly coherent, global shared memory, which is implemented across its physically distributed processor memories by a combination of operating system software and hardware support through the KSR ALLCACHE search engine, KSR. (1994). The latter is a particularly distinctive feature of this machine. Each processor is a 20 MHz super-scalar RISC chip with a peak 64 bit floating point performance of 40 Mflop/s and 32 Mbyte of local memory. Therefore the KSR1-64 at Manchester has a theoretical peak performance of 2.5 Gflop/s and 2 Gbyte of memory. A Gflop/s (or gigaflop) is equivalent to 1,000 million floating point operations per second; a top end PC might manage 10 million.

Parallelisation strategies

The aim in parallelising the spatial interaction model is to rework the serial code in order that the computational load can be distributed over multiple processors. On the KSR this is done by adding compiler directives to the serial Fortran code, KSR. (1993). Of course this approach to parallel code development will only work well if the code begin ported is already implicitly suitable for concurrent computation. The spatial interaction model is an example of what might be regarded as an embarrassing parallel model that is well suited to both vector and parallel supercomputers without much algorithmic change. The principle areas of parallelisation in the model are in the vector computation of the A_i and B_j terms and in running the final model, in that the model's predictions for each of the destination zones are independent, they can be assigned to many different processors and thus performed concurrently.

Another design aspect concerns how to make good use of a parallel processor with distributed memory. Although the available memory was insufficient to store all three matrices (trips, cost and deterrence function), there was enough space to store one of them. It was considered best to store the matrix that minimised the amount of extra computation that was needed and the choice depended on whether the model was singly or doubly constrained. For the latter model, the deterrence function matrix was stored as it was repeatedly used in the iterative estimation of the A_i and B_j terms. This resulted in a program that required 928 Mbyte of memory but only increased the amount of computation by a factor of two compared with the full memory version. In the singly constrained model the cost matrix was stored.

In order to study the memory issues as well as the parallelism, two versions of the code were produced: a *semi-sparse* version where the most useful matrix is stored in full and a *sparse* version where none of the three matrices are stored. An important aim in parallelising the code is to minimise the amount of communication needed to move the data around the system and to evenly distribute the computation load between the processors. Therefore, it is important that the computational work is divided between the processors so that each processor works only on those contiguous section of the matrix stored in its local memory, thereby minimising the amount of non-local accesses. The spatial interaction model code is well suited to this form of data parallel computing. There are a series of double DO loops which can be allocated to different processors to achieve a high degree of parallelism. The same strategy was used to parallelise the sparse version of the code. The difference between the codes is the amount of extra work needed to calculate the cost and deterrence function matrices on demand, which adds work to the loops, but it does not affect the parallelism other than remove the data location and storage issues since a copy of the trip data held in sparse form can be held on each processor. The memory requirements of the sparse version also make it feasible to run it on small numbers of processors.

Two forms of model are run: an *Origin Constrained* model and a *Doubly Constrained model.* In both cases the parameter(s) were estimated by *Maximum Likelihood Methods.* The timing results for the sparse and the semi-sparse version of the code on various numbers of processors and on a Sunsparc 10/41 workstation are shown in Table 1. The doubly constrained version also required 29 model evaluations but with 674 iterations to calculate the A*i* and the B*j* values. The timing results are shown in Table 2.

Table 1 **Results for the origin constrained model**

Number of Processors	Data storage version	Computation Time (minutes)
8	Sparse	57
16	Sparse	29
24	Sparse	19
28	Spares	17
32	Sparse	15
40	Sparse	12
48	Sparse	10
24 *	Semi Sparse	16
28	Semi Sparse	14
32	Semi Sparse	12
40	Semi Sparse	9
48	Semi Sparse	7
Sunsparc 10/41 Workstation	Sparse	1060

Note:

* the semi-sparse data storage version needed at least 24 processors before it could be run, due to its memory requirements.

Table 2 **Results for the doubly constrained model**

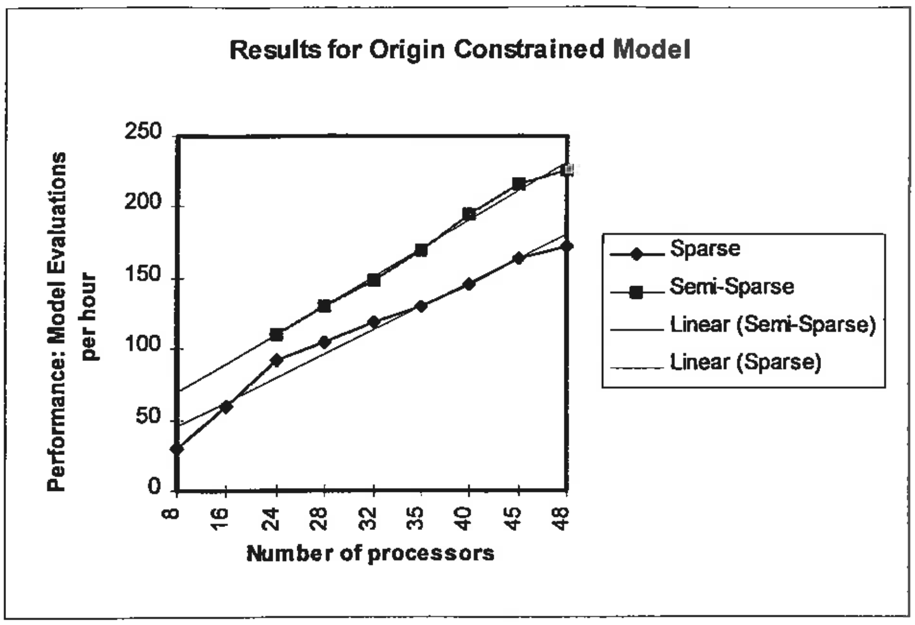
Number of Processors	Data storage version	Computation Time (minutes)
8	Sparse	2090
16	Sparse	1097
24	Sparse	730
32	Sparse	510
40	Sparse	432
24	Semi Sparse	78
28	Semi Sparse	60
32	Semi Sparse	47
40	Semi Sparse	36
48	Semi Sparse	29
Sunsparc 10/41 Workstation	Sparse	158502

Parallelism

It is clear that the parallel code runs considerably faster than the serial code. In fact with 48 processors the code runs 138 time faster for the singly constrained model and 530 times faster for the doubly constrained, when compared to its speed on a Sunsparc 10/41 workstation. This is an extremely large speed-up , however, it should be remembered that the performance gain for the semi-sparse code is partly due to the concurrency and partly due to the use of memory (in this case almost one Gbyte) to reduce the amount of computation that was done. It is noted that when George (1993) reported the results of a data parallel version of a retail spatial interaction model used in an optimisation problem on the Thinking Machines CM-200 parallel machine at Edinburgh, she managed speed up factor of 2260 times. However, this was a different model, run on a very different SIMD architecture, and was run on a smaller size of problem with all the matrices in memory.

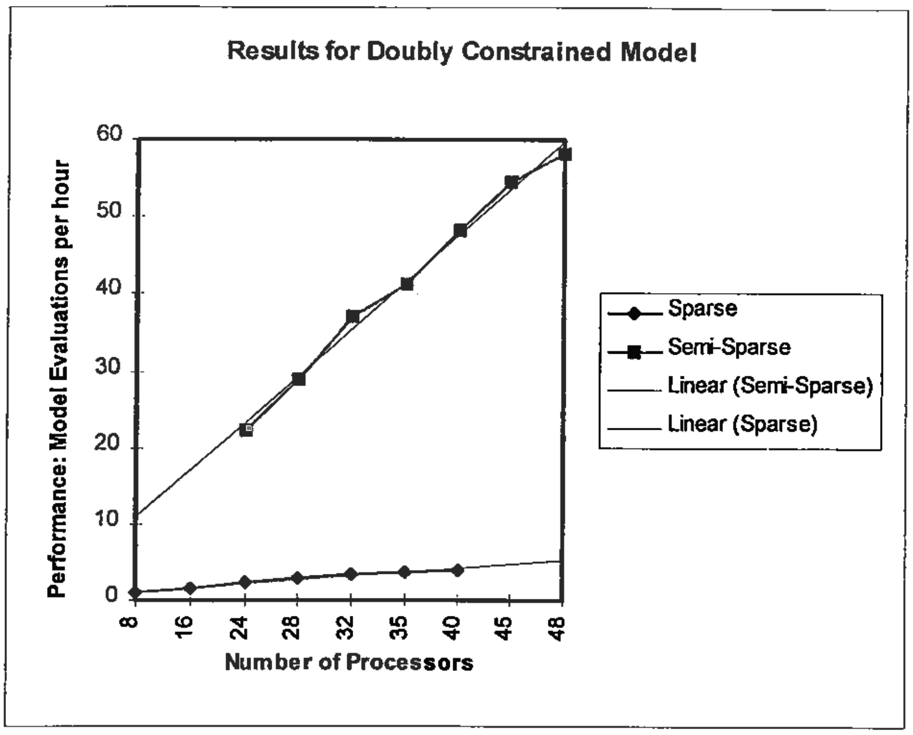
All versions of the code demonstrate good scaling and parallel efficiency. Figures 1 and 2 show the performance of the code against the number of processors used. Performance here is expressed in terms of the number of model evaluations per hour of compute time. This is useful because in both calibration and optimisation applications the models could be run a few hundred or even several thousand times. It might also be regarded as a measure of the amount of science being performed. The main reasons for this scalability reflect the large amount of computation which is present in the problem which can be performed in parallel. The parallelism is also moderately coarse grain which results in high levels of parallel efficiency. In the origin constrained case the semi sparse version is faster than the sparse by only a small factor, whereas in the doubly constrained version the semi-sparse sparse case is an order of magnitude better than the sparse. This is due to the massive saving in computation that can be achieved in the doubly constrained model by storing the deterrence function matrix and demonstrates the additional benefits of having a large memory space, which is another feature of parallel computers. The computational saving in the semi-sparse sparse version of the origin constrained case is much lower as there is no iterative calculation of the deterrence function matrix, although a noticeable performance gain can still be achieved by exploiting the KSR's large memory.

Figure 1



Results for Origin Constrained Model
Performance: Model Evaluations per hour
250
200
150
100
50
0
8
16
24
28
32
35
40
45
48
Number of processors
Sparse
Semi-Sparse
Linear (Semi-Sparse)
Linear (Sparse)

Figure 2


Results for Doubly Constrained Model
Performance: Model Evaluations per hour
0
10
20
30
40
50
60
8
16
24
28
32
35
40
45
48
Number of Processors
Sparse
Semi-Sparse
Linear (Semi-Sparse)
Linear (Sparse)

A two parameter variant of the standard entropy model was also investigated. This is a more sophisticated model that requires more computation during each model evaluation and more model evaluations. The results for the singly constrained case are shown in Table 3 and the doubly constrained in Table 4. Figure 3 shows the performance for both the singly and doubly constrained versions against the number of processors used. As with the previous model the code scales well although it can be seen that the performance obtained is less than before; this is due to the extra computation needed for each model evaluation. The number of model evaluations required has also increased from 29 to 69 in the singly constrained case and from 29 to 56 in the doubly constrained case. The two parameter results display the same properties with respect to parallelism and performance as the standard model and demonstrates that more complex models can also benefit from the generic parallelism inherent in the spatial interaction model. It would be interesting in the future to see how well other parallel supercomputers can cope with the same problem.

Conclusions

The paper describes the development of a parallel spatial interaction model and its application to what is probably the largest flow matrix yet modelled. The results are interesting both in their own right and also as a means of increasing awareness of what high performance computing (HPC), in general and parallel super computing in particular, can offer. It should be noted that this year's models of parallel supercomputers offer speed-ups of between 20 and 70 times above that of the KSR machine used here. Suddenly geographical modellers are being freed from the computation constraints that have existed ever since the early years of the quantitative revolution. Before long there will probably be no relevant computation or memory constraints to hinder any of their modelling ambitions. As Openshaw (1994b) points out, this could be the dawn of a major new era in geography, the beginning of what can be termed "computational geography"; defined as new computationally intensive ways of doing geography. HPC is not just a means of making old models run two or three orders of magnitude faster but much more significantly it creates an opportunity to solve many previously unsolvable problems, to develop new computationally based technologies and to create entire new ways of thinking about and doing all kinds of geography not all of which were

Table 3 **Results for the two parameter origin constrained version**

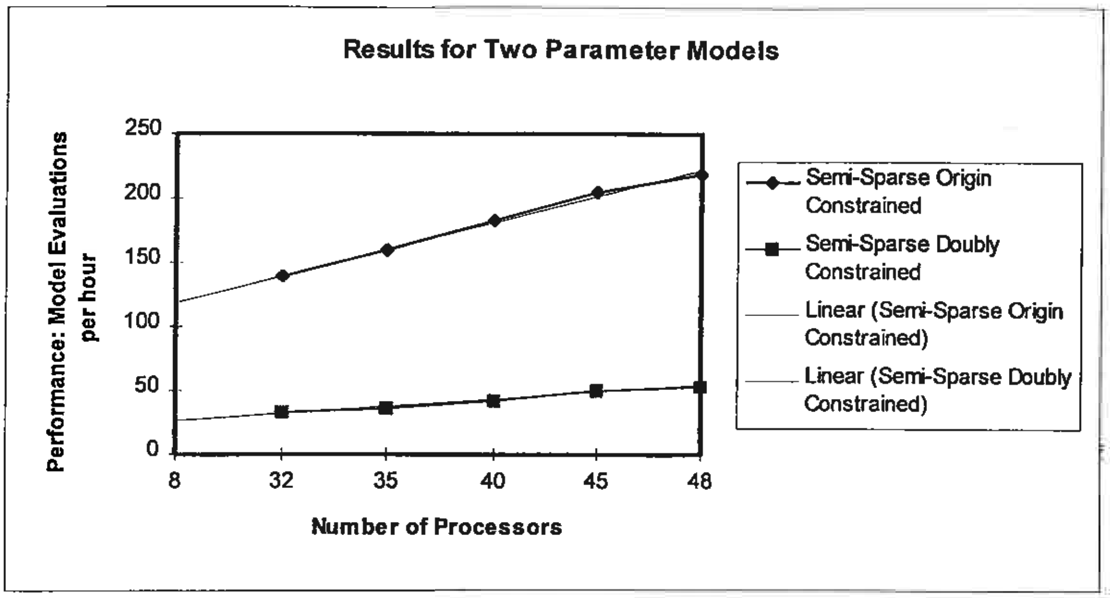
Number of processors	Data storage Version	Computation Time (in minutes)
32	Semi Sparse	30
35	Semi Sparse	26
40	Semi Sparse	23
45	Semi Sparse	20
48	Semi Sparse	19
Sunsparc 10/41 Workstation	Sparse	2610

Table 4 **Results for the two parameter doubly constrained version**

Number of processors	Data storage Version	Computation Time (minutes)
32	Semi Sparse	103
35	Semi Sparse	92
40	Semi Sparse	79
45	Semi Sparse	66
48	Semi Sparse	63
Sunsparc 10/41 Workstation	Sparse	331806 *

* estimated

Figure 3



Results for Two Parameter Models
Performance: Model Evaluations per hour
250
200
150
100
50
0
8
32
35
40
45
48
Number of Processors
Semi-Sparse Origin Constrained
Semi-Sparse Doubly Constrained
Linear (Semi-Sparse Origin Constrained)
Linear (Semi-Sparse Doubly Constrained)

previously quantitative. Hopefully the results reported here will help stimulate awareness and assist this process.

Acknowledgement

Some of this research was funded by the EPSRC. The support of the Centre for Novel Computing at the University of Manchester is also gratefully acknowledged.

References

Birkin, M., Clarke,, and George F. (1995), 'The use of parallel computers to solve nonlinear spatial optimisation problems', *Environment and Planning A* forthcoming).

Braunl, T., 1993 *Parallel Programming: an introduction* Prentice Hall. New York

Harris B. (1985), 'Some notes on parallel computing with special reference to transportation and land use modelling', *Environment and Planning A*, 17, 1275-1278

KSR (1993), *KSR FORTRAN Programming*, Kendall Square Research. Waltham, MA

KSR. (1994) *KSR Parallel Programming Guide,* Kendall Square Research, Waltham, MA,

Openshaw, S. (1987), 'Some applications of supercomputers in urban and regional analysis and modelling', *Environment and Planning A*, 19, 853-860

Openshaw, S. (1994a) 'Some geographical applications of high performance computing', *Working Paper 94/18*, School of Geography, Leeds University

Openshaw, S. (1994b) 'Computational Human Geography: towards a research agenda', *Environment and Planning A, 26, 499-505*