TOWARDS A REALISTIC PERFORMANCE MODEL FOR NETWORKS OF HETEROGENEOUS COMPUTERS

Alexey Lastovetsky
and John Twamley

University College Dublin, Belfield, Dublin 4, Ireland
Alexey.Lastovetsky@ucd.ie, John.Twamley@ucd.ie

Abstract This paper presents experimental work undertaken towards the development of a realistic performance model for non-dedicated networks of heterogeneous computers. Unlike traditional models, it is aimed at parallel computing on common networks of computers and distributed computing on global networks. It takes into account the effect of paging and differences in the level of integration into the network of computers, with the inevitable fluctuation in the workloads of computers in such networks. Some preliminary experimental work undertaken in support of the development of the model is briefly discussed. Based on the results of these experiments some key parameters of such a model are proposed.

Keywords: performance models, heterogeneous networks, global networks, parallel computing, distributed computing

1. Introduction

Networks of computers (NOCs) are the architecture increasingly used for high performance computing. Over the last 10 years many applications have been written to efficiently solve problems on local and global NOCs. Computers on this type of network are heterogeneous in that they have different architectures. Programmers seeking to write parallel applications for heterogeneous networks, where the goal is to optimize the execution time of that application, must partition and distribute computations and data unevenly to the computers on which the application will run, in proportion to their relative speeds. The absolute speed of a computer may be defined as the number of computational units of a benchmark or test application, executed in a given time.

Relative computer speeds. Some performance model must be employed to predict and represent the relative speeds of the computers. Performance mod-

els employed have normally described relative computer speeds using constant positive numbers. Each computers speed is represented using the ratio of its speed against the other computers. Various methods have been employed to obtain these ratios. Early performance models used the MIPS (millions of instructions per second) or MFLOPS (millions of fbating point operations per second) rates as the basis for the prediction of the comparative speed of computers. Another approach is to run the same benchmark application on each computer and compare their relative speeds. Relative computer speeds were used in [1]. as an aid to partitioning 2D grids representing science and engineering problems and in [2] in partitioning Matrix Multiplication problems. The use of real applications to obtain comparative computer speeds was used in the design of the mpC programming language [3].

Computer performance may be determined before execution, or dynamically, at run time. The best know system for use by dynamic schedulers to predict the performance of networked computers is the Network Weather Service [4]. The Network Weather Service monitors the fraction of the CPU utilization available to a newly started process on networked computers and represents this using decimal numbers. Available physical memory is also monitored, as well as network conditions. However, no allowance is made for differences in applications, or for the fluctuations in workload observed in networked computers as discussed below. Some examples of dynamic global based scheduling solutions can be found in , [4],[5], [6] and [7].

In existing performance models these relative speeds are usually taken to be constant across a range of dataset sizes. This assumption is problematic; it has been shown that relative speeds can change with increased dataset sizes. If the computers on the network have significant differences in the size of each level of their memory hierarchies then their relative speeds will change as the size of the dataset used increases. The result of an experiment conducted by [8] using a serial application which multiplies two matrices is reproduced below. Table I shows the specifications of the computers used. Figure 1 illustrates the non-constant relative speeds exhibited as the application is run with increasing size datasets.

As the relative performance of the computers is not constant for all datasets, we cannot run a trial run of a task on each computer in the network with a small dataset, in order to accurately predict the relative performance of computers executing the task when run with a larger dataset. A model to accurately predict relative computer speeds for a particular task must reflect the fact that relative speeds will change as dataset sizes increase.

Non-dedicated networks and fluctuations in speed. Computers on NOCs used for high performance computing will experience fluctuations in their workload. This is a disadvantage of integration into the network. The higher the level of integration the greater the level of fluctuations observed. This chang-

Table 1. Specifications of 4 computers with different memory hierarchies

Machine. Name	$Architecture$	$CpuMHz$	$MainMemory$	$Cache(KB)$
Comp1	Linux 2.4.18-3 i686	499	513960	512
Comp2	SunOS 5.8 Ultra 5.10	440	409600	2048
Comp3	Linux 2.4.18-3 i686	996	254576	256
Comp4	Linux 2.4.18-3i686	499	126176	312

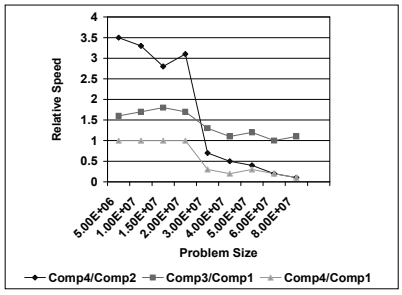

Figure 1. Non-constant relative speeds for computers with different memory hierarchies

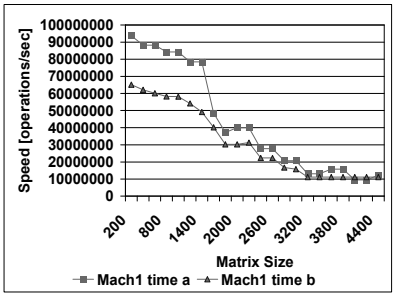

Figure 2. Differences in speed on a computer with a high level of network integration

ing transient load will cause a fluctuation in the speed of computers on the network, in that the speed of a computer will vary when measured at different times while executing the same task.

Performance models have up to this point concentrated on partitioning data and computations on dedicated computer systems; they have not considered the issue of load fluctuation as a factor in the prediction of the computational

speed of a networked machine. This fluctuation in workload on a computer is unpredictable; therefore it is impossible to make an accurate prediction of the speed of the computer for a particular application. A model for a non-dedicated heterogeneous network must take account of this. A realistic approach is to determine that the speed of a computer for an application will be between an upper and lower level. In general the less integrated a computer is in a network the more accurate a speed prediction is likely to be. Figure 2 illustrates fluctuations in the speed of a server computer with a high level of network integration, running a serial matrix multiplication application. Two performance curves are shown, time a and time b, generated at different times. As can be seen there is a considerable difference in the speed of the computer, which can be explained by fluctuations in workload.

In addition to normal routine user or network loads as described above, there is also the possibility that a networked computer may be performing some heavy computational task previously assigned by a parallel application or scheduled by a grid system. A model should distinguish between the two types of workload, reflecting any decrease in computer speed resulting from heavy computational workloads.

Paging and performance degradation. Paging occurs when the dataset size of a task exceeds the size of available physical memory. A high level of paging will lead to a significant decline in a computers speed. In the past, the partitioning of parallel applications for dedicated systems was normally done to avoid paging. As a result, performance models have mostly assumed a linear relationship between problem size and execution time. However it is necessary to understand the effect of paging on the speed of a computer executing a task. The overhead of paging may be less than the communication overhead involved in dividing a task. In addition it may be necessary to remotely solve a task which cannot fit into the physical memory of any computers registered to solve the problem.

While it is desirable to fit all of the data into main memory, there was a need to conduct some research into the development of a model which does attempt to address performance decline due to paging [9]. However, this model assumes carefully written applications, on dedicated computer systems only. A realistic performance model for a NOC, to be useful in partitioning applications when subtasks may exceed the physical memory of available computers, must seek to model performance decline due to paging.

A new performance model for non-dedicated heterogeneous networks. Previous performance models provided good results for tasks which did not exceed physical memory on dedicated computer systems. However they are not likely to provide optimum results for scheduling subtasks which may exceed the memory of available computers on NOCs, where computers have different sizes at each level of their memory hierarchy. A realistic performance model

for NOCs must take into account that relative computer speeds will not be constant, it may not be possible to accurately predict the speed of a computer for a particular application due to fluctuations in its workload and the application may not be suitable for partitioning to prevent paging. In addition, constraints on available computers or non-optimization in the design of the application may also lead to a situation where the avoidance of heavy paging is impossible.

The eventual goal of this research is develop a performance model, capable of predicting the speed of a computer on a NOC, for a particular task. The model should take into account non-constant computer speeds, fluctuations in workload and paging, with an acceptable level of efficiency and accuracy. It should provide for improved execution times over existing models for parallel applications on NOCs, when used as a basis for determining computer speeds when allocating subtasks to available computers.

To the best of our knowledge there has been no experimental study conducted providing detailed data on how the speed of computers for a range of tasks on a non-dedicated heterogeneous network changes for increased datasets. In addition no measurements have been taken as to the effect of fluctuations in workload on the speed of computers in this type of network. Based on some basic preliminary experiments we have some ideas of the characteristics of the performance model. To prove that the model is applicable to a wide range of applications it is necessary to carry out extensive experiments using a wide range of applications, run on different computers.

Part 2 of this paper describes the initial requirements of a model. Part 3 describes the experimental study. It details the types of applications chosen as providing a good basis for inclusion in a representative set of tasks and the computers chosen to demonstrate different levels of network integration and specifications. Part 4 presents some preliminary results of experiments on our set of tasks, examining the shape of the curves generated and the effect that a varying load on non-dedicated computers has on the level of accuracy of any performance prediction. Performance bands are illustrated to represent this fluctuation. In part 5 we look at some conclusions detailing the potential of our model to fulfill some of the requirements outlined in part 2 and mention further work to be carried out in the area.

2. The model and motivations

Based on preliminary experiments our assumptions of the main features of the model can be summarized as follows.

- In order to take account of paging the model proposes to use a function to represent the performance of each individual computer, absolute speed against the size of the problem.

- The efficiency of the model is crucial. Computational and communication overheads for the building and maintenance of this model should not lead to unacceptable degradation in the overall performance. This can be achieved by the specification of particular cases, which may be applied where there is no loss of accuracy.
- The speeds of integrated computers on NOCs should be characterized by bands of curves and not by single curves. This reflects the reality that we cannot determine speed with absolute certainty, due to the unpredictable nature of workload fluctuations, but may be able to predict likely upper and lower speeds, giving a more realistic level of accuracy.
- The efficiency of the model might be increased by classifying computers by their level of network integration. Different computers may be treated differently to provide accuracy with good efficiency.
- The model must be effective for computers which are already executing a significant heavy computational workload. Computers in a network may be required to carry out more than one parallel or distributed computing task simultaneously.
- The model should be generic, applicable to both distributed computing scheduling and parallel computing partitioning tasks.

3. Experimental Study

In view of the initial requirements of the model as stated above, the following experimental strategy was followed.

- Experiments were conducted with tasks ranging from those with efficient memory reference patterns, to those with very inefficient reference patterns, in order to include tasks with wide differences in the rate of decline of the performance curves generated with increasing datasets.
- On examination of the generated performance curves, a number of tasks were selected as a representative set. This set provided a basis for the possible classification of tasks as approximations of one of this set, based on the rate of decline of its performance curve.
- By repeatedly running the set of tasks on computers at different times it was possible to determine the effect of a computers level of network integration on the speed of the computer. The fluctuations in speed were modelled as a performance band.
- The performance bands for our applications were examined to determine how a consideration of the level of network integration of the computer might lead to their more efficient generation.

Table 2. Computers and Specifi cations

Machine. Name	$Architecture$	Cpu MHz	$Memory$	$Cache(KB)$
Mach1	Linux 2.4.20-20.9bigmem i686	2783	7933500	512
Mach2	SunOS 5.8 Ultra 5.10	440	4096000	2048
Mach3	Windows XP	3000	1030388	512
Mach4	Linux 2.4.7-10i686	730	254524	256

- Some experiments were conducted with applications run on computers already executing a significant computational load in order to examine the effect on the characteristics of the performance band.

The applications chosen included those which displayed both the most gradual and the most severe decline in the performance curve generated with increasing datasets. The applications chosen were.

- ArrayOpsF. Arithmetic operations on 3 arrays of data, accessing memory in a contiguous manner, with an efficient memory referencing pattern and use of cache.
- TreeTraverse. Recursively traverses up a binary tree in a post-order manner. The contents of an array of integers are placed into a binary tree, resulting in non-contiguous storage of an equivalent number of nodes, each containing storage for an integer and a pointer to a child node. Memory is accessed in a random manner, with no benefits derived from caching.
- MatrixMultATLAS. An efficient implementation of matrix multiplication utilizing the cblas_dgemm BLAS routine, optimized using ATLAS.
- MatrixMult. A naive serial implementation of the multiplication of 2 N*N dense matrices with the result placed in a third matrix. Memory will be accessed in a non efficient manner with an increasing number of page faults over larger problem sizes resulting in heavy IO.

Table II provides the specifications of the computers used in our experiments. They were chosen to provide varying specifications and levels of network integration. They are representative of the range of computers typically found on a NOC.

Mach1 has a very high level of network integration. It is a computer science departmental server running NFS and NIS, as well as web and database servers, with a very high level of use and multiple users. Mach2 has a medium level of

network integration, running database server and a web server. Mach3 runs p2p file sharing software and is connected up to the college LAN using Novelle, which provides a low level of integration. Mach4 is a networked machine, but provides no network services. It was anticipated that a high level of network integration would lead to a wide fluctuation in the speed of a computer over time.

4. Preliminary Experimental Results

4.1 Characteristics of performance curves

For each of the 4 applications, memory was obtained using malloc and gcc optimization -03 was used to compile the applications. It was considered that efficient memory referencing patterns would lead to a less steep decline in the speed of the application as problem sizes increased.

Figure 3 shows ArrayOpsF run on 4 computers. The performance curve generated by Mach1 is close to constant up to the maximum problem size capable of being solved. There is no sudden performance decline due to paging. Mach2, although its speed is significantly slower, exhibits the same constant characteristic. Mach3 displays the best performance until there is a sharp decline in speed, when paging causes the speed to dramatically decrease. A similar pattern is observed on Mach4, although the initial speed is slower than on Mach3, and the decline due to paging happens at a much smaller problem size. With the efficient memory access patterns and use of cache, the shape of the performance curve on all computers, except where paging causes a sudden decline in performance is close to constant. Figure 4 illustrates the execution speeds for 4 computers running TreeTraverse. Mach3 and Mach4 page heavily when their physical memory is exhausted. Mach1 levels off and becomes close to constant at problem sizes above 30 million nodes. Mach2 displays a slowly declining linear function. Figure 5 shows the generated curve for MatrixMultATLAS on our 3 UNIX clones. Mach1 and Mach2 are again close to constant for any significant problem size, Mach4 displays the same characteristic, but there is a decline in performance as paging occurs. Figure 6, because of the naive implementation of the application, was expected to show the most dramatic decrease in execution speed as problem sizes increased. Mach1 and Mach2 displayed a steady decline in speed with increasing problem sizes. Mach3 provided the fastest rate of execution over the whole of its problem range. Mach 4 also displayed a steady, reasonably linear decline. The experiments revealed that Mach1 and Mach2 are configured to avoid paging. This is typical of computers used as a main server. For applications designed to efficiently use cache memory, such computers may be characterized by a constant stepwise function, up to the point where the process crashes, probably because it tries to invoke a paging procedure, not allowed due to its configuration. This

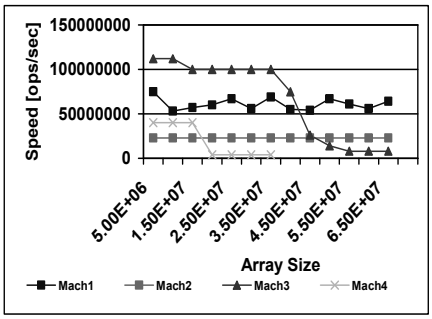

Figure 3. ArrayOpsF on 4 computers

is illustrated by examining the performance of Mach1 and Mach2, running MatrixMultAtlas, figure 5, with its efficient use of cache. Apart from small problem sizes, the speeds of both computers are constant. In contrast, both computers running MatrixMult, figure 6, show a decreasing speed with increasing problem sizes, levelling out only with large dataset sizes. This slowdown cannot be explained by paging, but is due to an increasing number of cache misses as the problem size increases. Non-paging computers, running tasks which reference memory in a random manner, not benefiting from caching, may also be characterized by a constant stepwise function. This is illustrated by figure 4, where we observe a close to constant speed for Mach1 and Mach 2, for all significant problem sizes, running TreeTraverse.

The efficiency of the overall performance model may be improved by differentiating between paging and non-paging computers, running applications designed to efficiently use cache, or applications which reference memory in a random manner. Although these 2 applications display contrasting levels of efficiency of cache use, they are similar in that the contribution of caching to performance is consistent for all problem sizes. Where computers are configured to prevent paging, the use of a constant stepwise function offers considerable efficiency gains.

Figure 7 illustrates the effect of cache use efficiency and paging on our performance curve, for each of our 4 applications, on a machine where heavy paging is permitted (Mach3 and Mach4). Performance is characterized using piecewise linear functions. The influence of caching and paging on the shape of the performance curve is illustrated for each application. The shape of the curve depends only on tasks and looks similar for both paging computers. For small dataset sizes, capable of being accommodated in physical memory, the slope of the curve is determined by the contribution of caching to performance. Where it is constant, as occurs with optimized tasks and tasks which do not

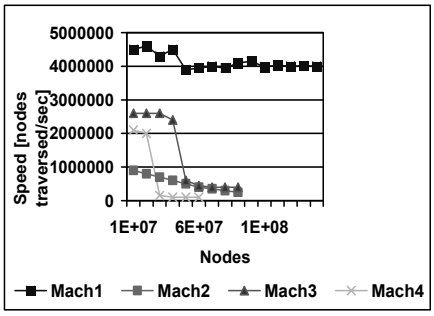

Figure 4. TreeTraverse on 4 computers

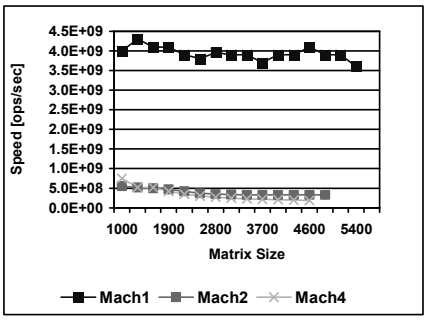

Figure 5. MatrixMultATLAS on 3 computers

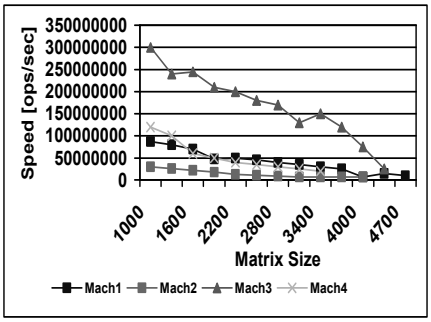

Figure 6. MatrixMult on 4 computers

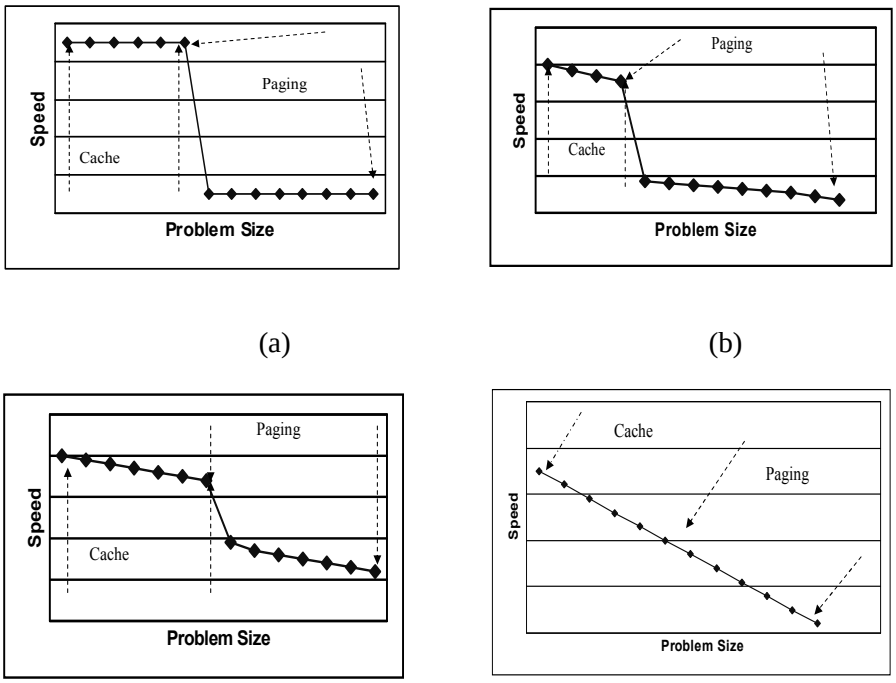

Figure 7. The effect of caching and paging in reducing the execution speed of each of our 4 applications, with performance characterized using piecewise linear functions .

benefit to any extent from caching, the slope of the curve will not deviate too significantly from the horizontal. This is apparent in figure 7 (a) and figure 7 (c), where the applications benefit from caching, and in figure 7 (b), where random memory reference patterns do not allow any benefit from caching. Figure 7 (d) illustrates an application where there is an increase in the proportion of cache misses, with increasing problem sizes. We notice a decline in speed for problem sizes smaller than those where paging occurs. Three of our applications, figure 7 (a), (b) and (c) show a sharp performance decline due to paging. However, the slope of the curve where this decline is occurring is not constant for all applications. Some tasks do not display this sudden decline, for example MatrixMult, figure 7 (d), where the slope of the line remains constant for the whole range of dataset sizes. Where a sudden performance decline occurs, the speed levels off again at the point where heavy paging is constantly occurring.

Our examination of figure 7 has shown that speed is not always constant up to the point where paging occurs, or when heavy paging is occurring. In addition, not all applications show a sudden stepwise reduction in speed when paging begins. As a result, the use of a stepwise linear function, as advocated by [9] cannot be used as a universal model for all applications on all computers. Our performance model will use a more general function, speed against the problem size, to allow for a more accurate prediction of speed for computers on NOCs.

In order to determine the effect of referencing different types of memory on the speed of a computer, ArrayOpsF, TreeTraverse and MatrixMult were run on the 4 computers with Automatic and static data. In some cases there was a difference in the execution speed of the applications, when compared to dynamic data of up to 15%, but this was not predictable. However, the shape of the curve retained the same basic characteristics as were apparent when dynamic data was used. All 4 applications were also run with gcc compiler optimization -01 on the four computers. As expected there was some slowdown, but again the shape of the curve was broadly similar to that obtained with optimization level -03.

4.2 Network integration and fluctuations in speed

Having considered the performance of the set of 4 applications on a range of computers typically found on NOCs, it is now necessary to examine the effect of a computers level of network integration on the accuracy of speed prediction. As stated above, a performance band instead of a curve was thought to offer an appropriate mechanism to model the realities of workload fluctuation. The performance band may be defined as the level of fluctuation which may occur in the performance of a computer executing a particular problem, due to

changes in load over time, expressed as a percentage of the maximum speed of execution for that problem.

The routine load on the computers was monitored and the 4 applications were run over a range of system loads. Figure 8 illustrates the performance band of our 4 computers running MatrixMult and TreeTraverse. Figure 8 (a) illustrates that the performance band for MatrixMult on Mach1 is around 40% for smaller problem sizes, narrowing to around 6% for larger problems. The decrease in absolute speed with increased problem sizes is due to the effects of caching, the computer is configured to avoid paging. For problem sizes with much longer execution times load fluctuations will be averaged out leading to a narrowing of the performance band. Figure 8 (b) shows the performance band for TreeTraverse on the same machine. The execution time is shorter than MatrixMult, so we don't see a narrowing of the band. In this case the performance band is in the order of 35% for it's whole range of problem sizes. Figure 8 (c) illustrates the performance band for Mach2 running MatrixMult. It can be seen that for smaller problem sizes the performance band is in the order of 15%. As might be expected the load fluctuation on this machine was less than that present on Mach1. Again the decline in absolute speed can be attributed to caching. The performance band displayed by Mach2 for Tree-Traverse as shown in figure 8 (d) is quite small, being reasonably constant at 8% for anything other than the smallest problem sizes. The performance band displayed by Mach3 for MatrixMult and TreeTraverse, shown in figure 8 (e) and (f), with it's low level of network integration, was not greater than around 5 to 7% even when there was heavy file sharing activity (it's effect seemed minimal). Mach4 displayed very little fluctuation for our 2 applications, with a performance band of around 3 to 5%, as might be expected from a virtually stand alone computer.

The size of the performance bands for Mach1 and Mach2, the computers with a high level of network integration, against time, are shown in figure 9. The influence of workload fluctuations on speed becomes less significant as the execution time increases. There is a close to linear decrease in the size of the performance band as the execution time increases.

We have noted that the fluctuation in speed observed for Mach1, the computer with the highest level of integration, is in the order of 40% for small problem sizes, declining to approximately 6% for the maximum problem size solvable on the computer. This level of fluctuation justifies the use of bands to model the speed of computers, instead of curves. The accuracy obtained with bands can still be within acceptable limits to be useful when predicting performance. The level of accuracy of the approximation is increased as the execution time increases with larger problem sizes. For computers with a low level of network integration such as Mach4 performance bands allow for a high level of accuracy in the prediction of performance.

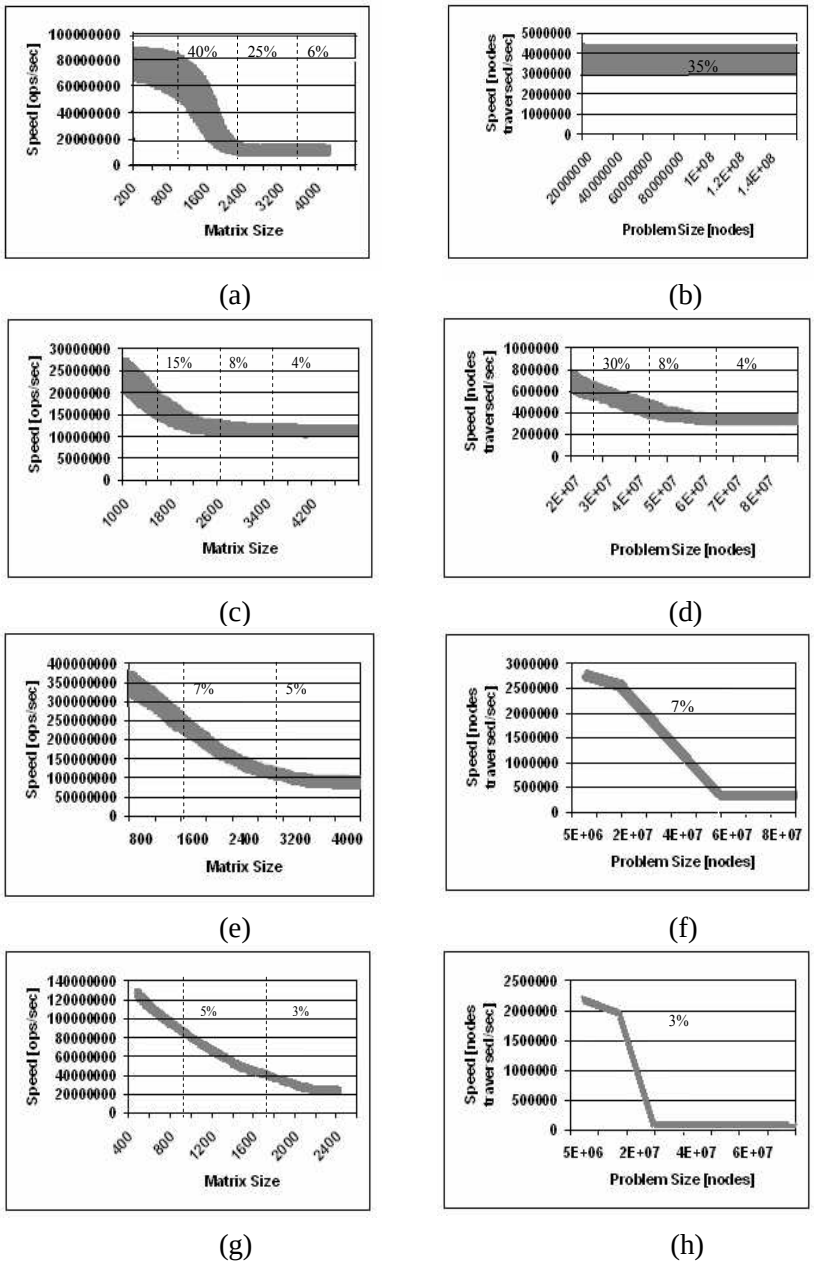

Figure 8. (a) Performance band for MatrixMult on Mach1, (b) Performance band for Tree-Traverse on Mach1, (c) Performance band for MatrixMult on Mach2 , (d) Performance band for TreeTraverse on Mach2, (e) Performance band for MatrixMult on Mach3, (f) Performance band for TreeTraverse on Mach3, (g) Performance band for MatrixMult on Mach4, (h) Performance band for TreeTraverse on Mach4.

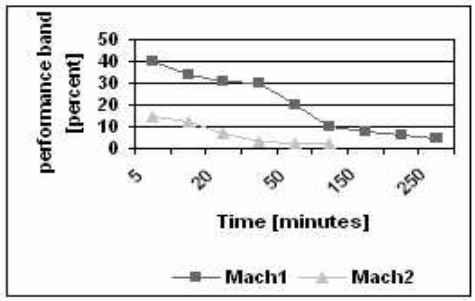

Figure 9. Performance band as a function of time on Mach1 and Mach2.

We have established that a different level for the accuracy of prediction is achievable for computers with high and low levels of integration. The efficiency of the model can be increased by taking this into account and treating computers differently. The number of measurements of speed against problem size required to achieve a realistic level of accuracy is inversely proportionate to the level of integration. A computer with a high level of integration will require very few measurements. A less integrated computer will require more measurements to reflect the higher level of accuracy achievable. The increased computational overhead required by computers with a low level of integration is offset by the likelihood of many of these computers being present on a typical network, with the prospect of utilizing a significant computational resource.

It is possible that computers displaying good performance, may be assigned, in addition to their routine workload, more than 1 heavy computational task. The performance band of Mach1, a computer with 4 processers, running Tree-Traverse, while already processing 4 substantial computational loads was constructed, simulating a situation where additional heavy loads are being executed on a computer. The effect of the heavy computational load on the performance band of the computer was determined. Figure 10 (a). shows the performance band while the workload was restricted to normal fluctuating routine computations. Figure 10 (b) shows the performance band generated while the additional 4 heavy computational loads were being run on the computer. Both the upper and lower level of speed is reduced but the overall width of the band remains relatively constant. It is thought likely that this lowering of the performance band occurs on all computers, executing all tasks, where the number of prior computational loads is greater than or equal to the number of processors present on the computer. More experimental work is required to investigate the effects of heavy existing workloads on performance predictions, particularly for single processor computers.

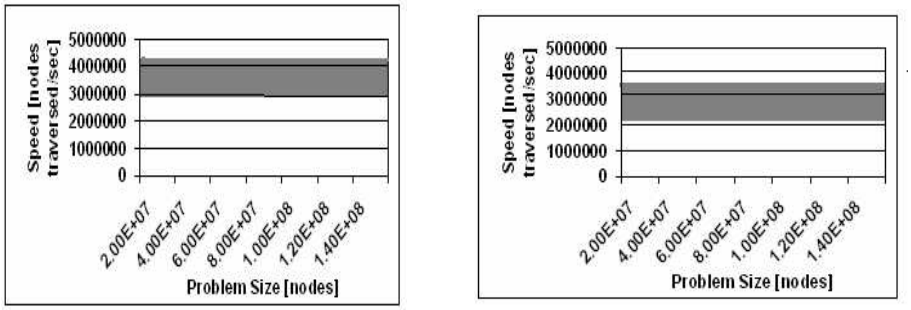

Figure 10. (a) Performance band for TreeTraverse on Mach1 with normal fluctuating workload. (b) Performance band for TreeTraverse on Mach1 with 4 prior major computational loads.

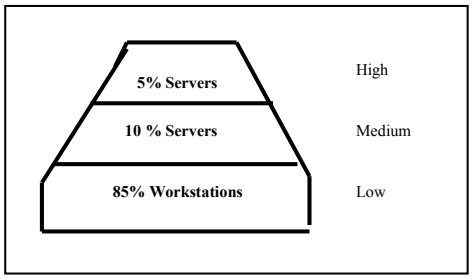

Figure 11. Typical avaliability of machines on a network classifi ed by network integration and function

5. Conclusions and further research

Based on the experimental results we can summarize the primary features necessary for inclusion in the model

- To address paging in parallel and high performance grid computing we use a function, the size of the problem against the absolute speed of the computer, rather than a constant or stepwise function. We have observed that the slope of the performance curve generated by different tasks may vary considerably. Also, for some tasks there is no sudden decline in speed when paging occurs.

- The model should be composed of sub-models, thereby increasing the efficiency of the overall model. The sub-model best suited to model the speed of a computer executing a particular application should be dependent on the computers level of network integration. Our approach is to seek to maximize efficiency where possible, without decreasing the accuracy of prediction. For example, a computer with a high level of integration will require less measurements to construct its performance band to an acceptable level of accuracy.

- The use of bands of curves, instead of a single curve, is the best method to model performance on NOCs. The width of the performance bands observed on computers with a high level of network integration is still sufficiently narrow to allow for an acceptable estimation of performance.

- Computers should be classified according to their level of network integration. Computers with a high level of network integration show large performance bands for small problem sizes, narrowing as the execution time increases. Initial observations indicate that the performance band may be approximated to a linear decrease over time, but more experimental work is needed to investigate this observation.

- The model should classify all computers on a network. Figure 11 shows the breakdown of computers on a typical NOC. The computational resource available when we combine our loosely integrated computers is significant and will usually exceed that of our highly integrated computers, therefore we must understand their performance characteristics if we are to efficiently exploit available resources.

- In our model we differentiate between computers that allow paging and computers that do not. Non-paging computers are typically servers with a heavy workload, offering a good relative performance. For carefully designed applications, efficiently utilizing cache, and for applications where memory is referenced randomly, the performance band of this

type of computer is linear, up to the maximum size of the problem solvable on the computer.

Some preliminary experimental work suggests its possible to reflect the fact that a computer is already engaged in a heavy computational task when seeking to predict its speed for a particular task. Initial work has suggested that while the upper and lower levels of speed are reduced, the width of the performance band remains close to constant. However, it may be possible to distinguish 2 types of prior computational load, each of which will have a different effect on the characteristics of the performance band. A heavy computational load with a small dataset may narrow the performance band, but not influence the dataset size at which heavy paging begins. In contrast, a task which utilizes a significant percentage of physical memory will cause paging to occur for even a small dataset. It is likely for some tasks, that paging may occur for even the smallest datasets. It may be possible, based on a measurement of CPU utilization and the degree of paging observed to predict the likely effect on the performance band. This would increase the efficiency of the model where heavy computational work is being executed on a computer. However, further work is required to investigate the feasibility of this approach.

More experimental study will be carried out to confirm the preliminary experimental results presented in this paper. The goal is to provide a practical performance model for NOCs enabling a realistic prediction of the speed of a computer for a particular task.

References

[1] Crandall, P. and M. Quinn. Problem Decomposition for Non-Uniformity and Processor Heterogenity. *Journal of the Brazillian Computer Society*, 2(1):13–23, 1995.

[2] Beaumount, O., Boudet, V., Rastello, F. and Y. Robert. Matrix Multiplication on Heterogeneous Platforms. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1033–1051, 2001.

[3] Arapov, D., Kalinov, A., Lastovetsky, A. and I. Ledovskih. A Language Approach to High Performance Computing on Heterogeneous Networks. *Parallel and Distributed Computing Practices*, 2(3):323-332, 1999.

[4] Wolski, R. Dynamic Forecasting Network Performance Using the Network Weather Service. *Cluster Computing*, vol 1:119–132, January 1998.

[5] Zaki, M.J., Lei, W. and Parthasarathy, S. Customized Dynamic Load Balancing for a Network of Workstations. *Journal of Parallel and Distributed Computing*, 43, 156-162, 1997.

[6] Berman, F. *High Performance schedulers. The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 1988.

[7] Vadhiyar, S. , Dongarra, J. and A. Yarkhan. GrADSolve - RPC for High Performance Computing on the Grid. *Euro-Par 2003, 9th International Euro-Par Conference, Proceedings*, Springer, LCNS 2790, 394-403, August 26-29, 2003

[8] Lastovetsky, A. and R. Reddy. Data Partitioning with a Realistic Model of Networks of Heterogeneous Computers. *Proceedings of the 18th international parallel and distributed*

processing symposium (IPDPS 2004), 26-30 April 2004, Santa Fe, New Mexico, USA. Computer Society Press.

[9] Drozowski. M. Out-of-Core Divisible Load Processing. *IEE Transactions on Parallel and Disbributed Computing*, 14(10):1048-1056, 2001.