A Layered Approach to Parallel Software Performance Prediction : A Case Study

E. Papaefstathiou, D.J. Kerbyson, G.R. Nudd

Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.
Email: stathis@dcs.warwick.ac.uk

ABSTRACT

An approach to the characterisation of parallel systems using a structured layered methodology is described here. The aim of this is to produce accurate performance predictions which maybe used to influence the choice of machines and investigate implementation trade-offs. The methodology described enables the separate characterisation of both application, and parallel machine to be developed independently but integrated though an intermediary layer encompassing mapping and parallelisation techniques. The layered approach enables characterisations which are modular, re-usable, and can be evaluated using analytical techniques. The approach is based upon methods introduced in Software Performance Engineering (SPE) and structural model decomposition but due to its modular nature, takes less time for development. A case study in image synthesis is considered in which factors from both the application and parallel system are investigated, including the accuracy of predictions, the parallelisation strategy, and scaling behaviour.

1. INTRODUCTION

One of the factors that has contributed to the limited acceptance of parallel systems from the mainstream community is the lack of efficient, user-friendly, methods and tools that enables accurate performance predictions to be made. A novel layered approach is described in this paper that provides performance predictions of parallel software before porting it [Nudd93, Papaefstathiou93]. This methodology is derived from the principles of structural model decomposition [Patel92] and Software Performance Engineering (SPE) [Smith90], which have been widely applied to sequential systems [Smith82, Weishar87], but are here applied to parallel systems.

A conventional modelling process starts by an examination of a computer system and the construction of a model of it. The measurement of the execution pattern, and the characterisation of the workload are required as input parameters to this model. A progressive refinement of this model is made by the comparison of real and predicted measurements until accurate results are yielded. Thus, such an approach is inadequate for predicting the performance of software which has not yet been ported since the workload cannot be measured.

The modelling approach used within SPE enables workload parameters to be determined without first implementing the software. SPE achieves this by complementing the conventional computer system model with a software execution model containing the key characteristics of the execution behaviour. The software execution model is constructed using workload scenarios, software design, execution environment, and resource usage information. It is evaluated by graph analysis algorithms to determine the workload parameters required for the system model.

However, SPE has a number of disadvantages that are particularly apparent for the performance prediction of parallel programs. These disadvantages include: a complicated and time consuming development procedure for the system model; analytical methods usually can not be used to evaluate the system model; the system model is software / hardware dependent and hence not re-usable.

The novel layered approach described here overcomes these limitations by the use of independent layers for each of the application, the parallelisation technique, and the hardware. The layered technique originates from work in model decomposition [Jain89, Patel92] whose main goal is to sub-divide the model into simpler models. The layered approach can be viewed as complimentary to SPE for the case of modelling parallel systems. The independence of the layers leads to:

- a minimisation of the time required to construct models
- wider applicability of analytical techniques for the models opposed to the models within SPE studies
- re-usable models which allow the easy experimentation of the performance of an application running on different hardware and using different parallelisation techniques.

In this paper the layered approach is first described in terms of its constituent layers namely the application layer, the sub-task layer, the parallelisation template layer and the hardware layer. An image synthesis application is used to illustrate the layered approach as a case study. The accuracy of the resulting characterisation is examined and it is shown how optimum values of parallelisation parameters can be obtained when experimenting with the resulting analytical model.

2. THE LAYERED APPROACH

The layered approach described here separates out the hardware and software systems through the use of a parallelisation template, Figure 1. This modular approach leads to readily re-usable models which can be interchanged in experimentation. For instance the performance predictions across parallelisation techniques can be compared for a particular application on a particular hardware. The layers used are listed below and are described in further detail in the following sub-sections:

- an application layer, which models the application in terms of a sequence of sub-tasks using a software execution graph notation (from SPE). Each node of the graph can be a sequential processing node, a user defined parallel sub-task, or a parallel processing generic (from a library).
- an application sub-task layer, which models the sequential part of every sub-task within an application that can be executed in parallel. The result of the evaluation of these models is fed to the parallel template layer.
- a parallelisation template layer, that identifies the resource contention caused by the characteristics of the algorithms.
- a hardware layer, which is responsible for characterising the communication, synchronisation, and contention of resources.

2.1. The Application Layer

The purpose of the application layer is to characterise the application in terms of a sequence of sub-tasks using a software execution graph. The graph notation includes control statements, hierarchical components, state identification nodes (lock-free, send-receive), and split nodes (concurrent processes), Figure 2. This provides a characterisation of the workload without the need of implementation and measurement. However, the accuracy of the resulting

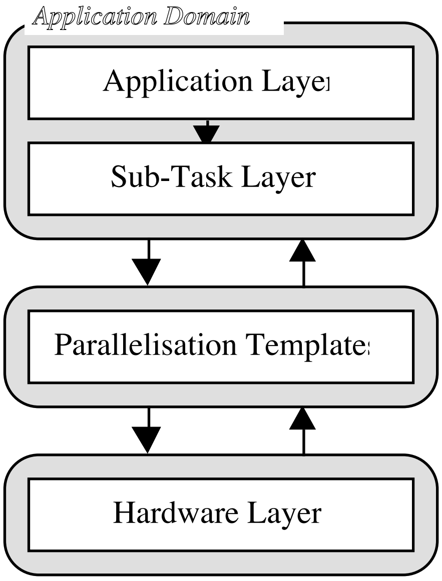

Figure 1 - The Layered Approach

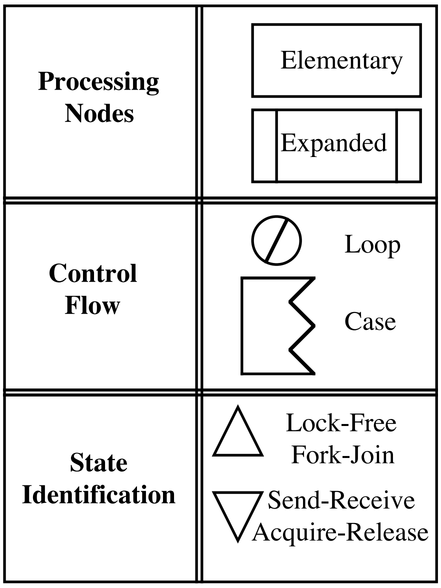

Figure 2 - Software Execution Graph Notation

predictions are dependent upon the techniques used to describe the software within the elements of the software execution graph.

The nodes of the software execution graph in the application layer consist of a number of elementary processing types, that are classified in the following categories:

- *Sequential Processing Nodes:* Tasks that can be performed sequentially.
- *User Defined Parallel Sub-Tasks:* These are defined in the lower sub-task application layer using the same graph execution notation.
- *Parallel Processing Generics:* These are frequently used pre-defined parallel sub-tasks which exist within a library. The formation of such a library is currently under investigation [Peps93].

2.2 Sub-Task Layer

In the sub-task layer application specific models are defined for each sub-task. The result of the evaluation of these models is the input to the parallel template layer. The sequential parts of the parallel template must be modelled and their response time determined.

Initially for each sub-task a software execution graph is constructed. The user must then identify the resource usage of each elementary node as defined in SPE. There are a number of ways in which each node of the software execution graph can be characterised, this is commonly referred to as a resource usage description. These range from a high level description considering sequential timing information or parametric characterisation [Hockney93], an intermediate level considering abstract machine descriptions [Saavedra89], down to a low level instruction frequency analysis [Zemerly93a]. The level at which the hardware is characterised is also dependent upon the choice of the resource usage description. In the case study below we consider a high level timing description, but work is currently in progress to analyse these different levels of resource usage.

2.3. Parallelisation Template Layer

The purpose of the parallelisation template layer is to identify the resource contention caused by the characteristics of the algorithm. It has been noted that a large number of computations fall into a surprisingly small number of prototypical structures [Gehringer88].

In many areas it is possible to identify a small set of algorithm structures that can be modelled using traditional modelling techniques [Allen86]. Many parallelisation techniques have been analysed in detail [Greenberg91, Rolia92].

In order to identify the most commonly used parallelisation techniques a classification scheme is required. The aim of such a classification is to determine different classes, on the basis of similarities, amongst different parallel algorithms. The classification enables the extraction of the most frequently used constructs and also the definition of application independent parameters. Proposed classification schemes include: the Cm* project [Gehringer88], the Basel Algorithm Classification Scheme (BACS) [Burkhart93], and the scheme proposed by Jamieson [Jamieson87]. By considering the most representative parallelisation templates it is possible to cover the majority of applications.

A parallelisation template also includes the mapping of the algorithm onto network topologies. The purpose of determining the algorithm mapping is to identify the communication and synchronisation patterns that are inputs to the hardware layer. For example, different mappings of the master-slave paradigm on a mesh topology result in different communication radius influencing communication contention.

2.4. The Hardware Layer

The hardware layer is responsible for the characterisation of communication, synchronisation, and contention. The information required for this layer can be organised into a hierarchical structure similar to the one used in the architecture characterisation tool in the PAWS project [Pease91]. The requirements of the hardware model are less complex than the system model in SPE, making the development and evaluation procedure easier.

The hardware model consists of static and dynamic performance parameters. Static parameters (e.g. number of processors) represent hardware factors that are not influenced by the run-time environment and can be determined either by benchmarking or configuration information. Dynamic parameters are influenced by the run-time environment and can be determined by analytical models or hardware characterisation, e.g. [Zemerly93b].

3. Case study: An Image Synthesis Application

The case study considered here is the computational core of an image synthesis application, based on ray tracing techniques, and has also been developed into a benchmark [Biersack93]. The target machine in this case study is a Parsytec SuperCluster containing 128 T800 transputers.

This application requires a scenery file as input, renders the image, and finally outputs the generated image. The synthesis problem scales in three dimensions: the complexity of the scene, the resolution of the image, and its quality. The scene contains 4^n triangles, the image resolution is $2^m x 2^m$ and k ray-path samples per pixel (an image quality factor). The metric of the benchmark is the number of samples calculated during a fixed time period of 1000 seconds.

The purpose of the application layer in the case study is to combine the three phases of the application, input, rendering, and output. These phases can be parallelised with different strategies and are combined into a software execution graph. By evaluating this graph the total execution time of the application can be predicted:

$$t_{total} = t_{input} + t_{render} + t_{output} \qquad (1)$$

In the sub-task layer the time required for the sequential part of each phase is evaluated. For the following analysis the rendering phase is examined in detailed. The parallelisation strategies used here belong to the class of image-space subdivision algorithms [Green91]. The sequential time in these algorithms is the ray tracing of a pixel. Figure 3 shows a high level

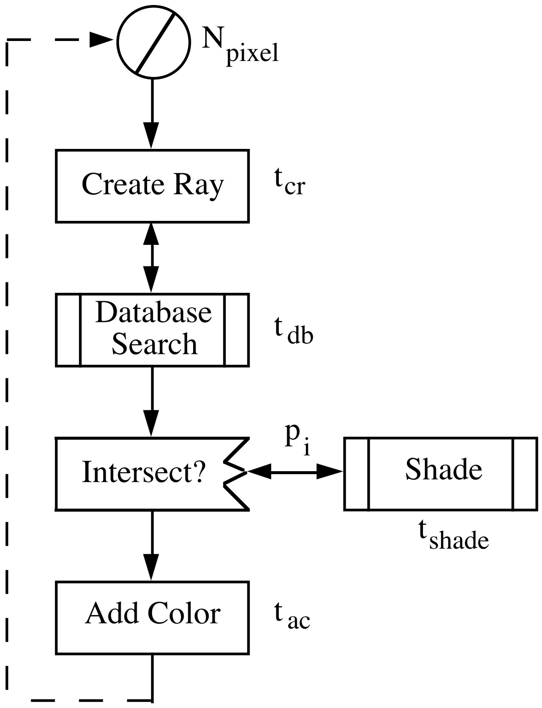

Figure 3 - Software Execution Graph for the Rendering Phase

execution graph for the calculation of one sample. From the evaluation of this graph the time required for one sample is:

$$t_{sample} = N_{pixel} \cdot \left(t_{cr} + t_{db} + p_i \cdot t_{shade} + t_{ac}\right) \quad (2)$$

where N_{pixel} is the number of pixels in the image region that a processor renders, t_{cr} the time required to create a new ray, t_{db} the time required for a database search to identify if the ray intersects an object, p_i the probability of the ray intersecting an object, t_{shade} the time required to shade the pixel with diffuse and specular reflections, and t_{ac} the time required to add colour to the pixel.

Two techniques have been considered for the parallelisation of this case study, namely the master-slave and the multiphase paradigms. Both of these methods have been applied in image synthesis applications [Holliman89, Coldsmith87]. In the master-slave paradigm, the master processor distributes work packets of rendering requests to slave processors. In the multiphase paradigm, each processor is assigned to a part of the image. When the rendering for one sample is completed, the master processor determines if there is sufficient time available to process further samples. Two parallelisation templates have been developed for these two cases.

The use of time as a resource usage descriptor simplifies the hardware layer but is complimented with a model for communication within the parallel system. This is based on a linear regression communication model [Bomans89] and has been extended for packet switching routing [Tron93]. More specifically:

$$t_{com}(l,d) = \begin{cases} \alpha + d \cdot \left(\beta + \tau \cdot (l + h)\right) & l \le p \\ \alpha + d \cdot (\beta + \tau \cdot p) + \left(\dfrac{l}{p} - 1\right) \cdot \left(\alpha_2 + \beta + \tau \cdot p\right) & l > p \end{cases} \quad (3)$$

where l is the message size, d distance between source and destination node, h is the size of the packet header, p the size of the packet, α is the time to prepare the message, β is the start up time, τ is the inverse of bandwidth, and α_2 is the time to prepare a packet.

4. RESULTS

A layered model has been developed for the above case study using Mathematica. This model is used to provide performance predictions whilst varying a number of application, hardware, and parallelisation parameters. Firstly the accuracy of the layered approach is examined. Two parallelisation templates are then considered and predictions calculated. The application parameters of problem size (e.g. number of image pixels, number of triangles) are also examined. Finally the scaling behaviour of the application is also investigated.

The first set of results aims to verify the accuracy of the model using the layered approach with real measurements produced from an implementation of the code using a Parsytec SuperCluster machine as a basis. The measured and predicted time per sample taken to generate an image (of size 4096 pixels with 256 triangles) is shown in Figure 4, and the %

difference between the two methods is shown in Figure 5. It can be seen that the predictions lie within an error bound of 25% of the true measurements, with the average error being nearer to 15%.

Performance predictions were obtained from the layered model using different parallelisation templates, namely those of a master-slave and multiphase paradigms. This was achieved by just interchanging the parallel template layer with the appropriate parallelisation technique without effecting other parts of the model. The predications obtained using both of these parallelisation techniques is shown in Figure 6. Here the relative performance is calculated as in Equation 4. It can be seen that the master-slave parallelisation technique outperforms the multiphase technique except in the case of a very small processor grid. This is as expected due to the dynamic load balancing effect of the master-slave technique.

$$P_{relative} = \left(1 - \frac{t_{master\_slave}}{t_{multiphase}}\right) \times 100 \qquad (4)$$

The effect of the application parameter of the number of triangles is examined in Figure 7. This shows the average predicted time spent to deal with one triangle across a range in the required number of triangles. The predicted time per triangle here is taken as the total time taken (e.g. Figure 4) divided by the number of triangles. It can be seen that as the number of triangles increases, the effective time per triangle decreases but levels out. For small numbers of triangles the effect of communication overhead is large with respect to the processing time, whereas for a large number of triangles this overhead is not significant.

The scaling behaviour of the application is examined in Figures 8 and 9. By examining the predicted speedup for between 20 and 200 processors it can be seen that a maximum speedup may be obtained on approximately 140 processors (with a speedup of 80). Above this size the speedup decreases due to excessive contention between the master and slaves. The effect of increased master-slave queuing time, and also communication time, can be seen in Figure 9.

SUMMARY

In this paper we have described an approach for the characterisation of applications on parallel machines for performance prediction without the need of implementation. The approach is structured into four layers, two dealing with the application, one encompassing parallelisation techniques, and the last to characterise the hardware. These layers are independent and re-usable such that different machine characterisations or parallelisation techniques may be interchanged.

A case study of image synthesis is used to illustrate this approach and a model has been developed using Mathematica. Predictions have been obtained for varying a number of application dependent parameters (e.g. problem size) and hardware parameters (e.g. number of processors). These predictions have been compared with actual measurements and show an average accuracy to within 15%. In addition, the effect of using different parallelisation strategies, and the scaling behaviour of the application has also been examined.

A number of refinements to the layered approach are in progress. These include an investigation into: the effects of different levels of resource usage (e.g. low-level instruction based, high-level abstract machine models) and their resulting accuracy; the integration of conventional modelling techniques (e.g. queuing networks) within the layered framework where applicable. The development of a characterisation tool is also under way, that will enable the characterisation of the application, the parallelisation templates, and the hardware to be undertaken within a homogeneous language.

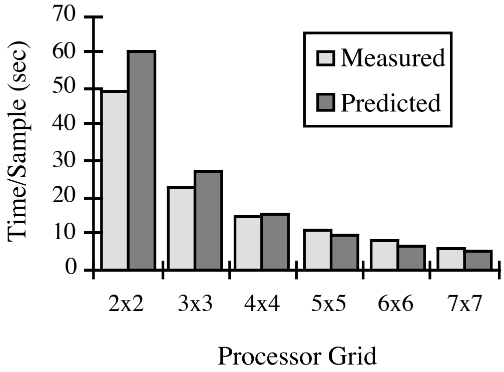

Figure 4 - Measured vs Predicted Time, Master-Slave (256 triangles, 4096 pixels)

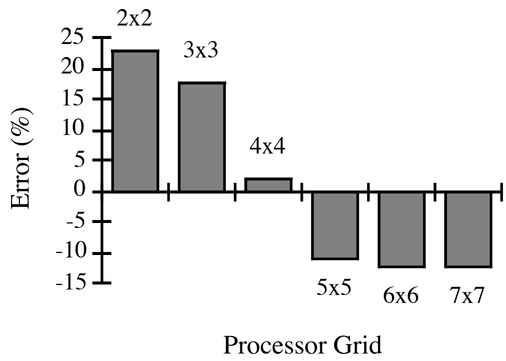

Figure 5 - Accuracy of the Layered Approach, Master-Slave (256 triangles, 4096 pixels)

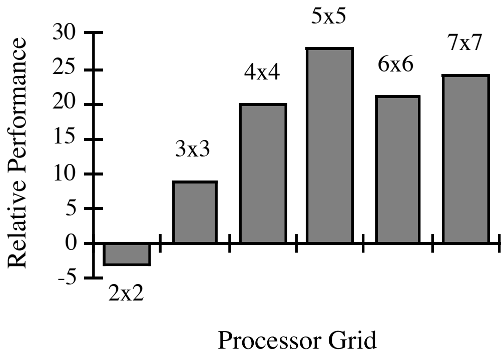

Figure 6 - Master-Slave vs Multiphase, Predicted (256 triangles, 4096 pixels)

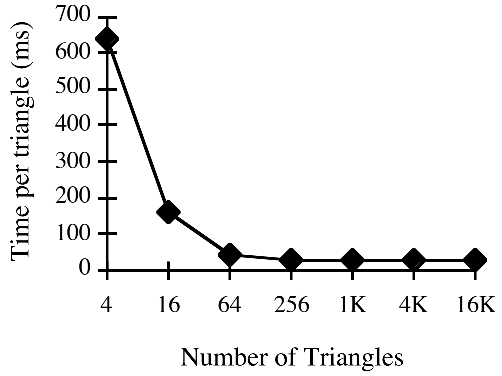

Figure 7 - Effects of Scene Complexity, Predicted (36 Processors, 4096 pixels)

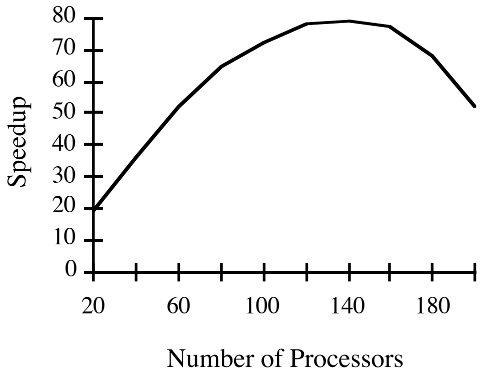

Figure 8 - Speedup of the Master-Slave, Predicted (256 triangles, 4096 pixels)

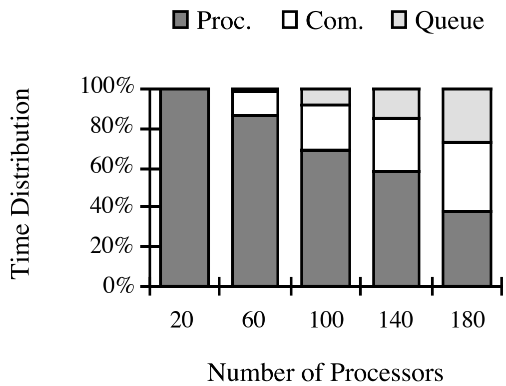

Figure 9 - Time per Function, Master-Slave, Predicted (256 triangles, 4096 pixels)

ACKNOWLEDGEMENTS

This work is funded in part by ESPRIT contracts 6942 - Performance Evaluation of Parallel Systems (PEPS) and 6173 - Design by Simulation and Rendering on Parallel Architectures (DESIRE).

REFERENCES

[Allen86] Allen, J., "Plenary Address," in *IEEE Workshop on VLSI Signal Processing,* 1986.

[Biersack93] Biersack, A and Hodicke, R., "The MEASURE Image Synthesis Benchmark," in *Proc. Performance Evaluation of Parallel Systems (PEPS'93)*, Coventry, U.K., pp. 150-157, 1993.

[Bomans89] Bomans, I. and Roose, D., "Benchmarking the iPSC/2 HyperCube Multiprocessor," *Concurrency: Practice and Experience*, 1989.

[Burkhart93] Burkhart, H., Korn, C.F., Gutzwiller, S., Ohnacker, P., and Wasel, S., "BACS: Basel Algorithm Classification Scheme," Tech. Rep. 93-3, University of Basel, Switzeland, 1993.

[Gehringer88] Gehringer, E.F., Siewiorek, D.P., and Segall, Z., *Parallel Processing: The Cm* Experience*. Digital Press, 1988.

[Goldsmith85] Goldsmith, J. and Salmon, J., *A Ray Tracing System for the Hypercube*, Caltech Concurrent Computing Project Memorandum HM154, California Institute of Technology, 1985.

[Green91] Green, S., *Parallel Processing for Computer Graphics*, Pitman London, 1991.

[Greenberg91] Greenberg, G.A. and Wright, E.P., "Design and Analysis of Master/Slave Multiprocessors," *IEEE Transactions on Computers*, vol. 40, no. 8, pp. 963-976, August 1991.

[Hockney93] Hockney, R.W., "Performance Parameters and Benchmarking of Supercomputers," in *Computer Benchmarks,* Dongarra, J.J., and Gentzsch, W., Eds., Elsevier Science Publishers B.V., pp. 41-64, 1993.

[Holliman89] Holliman, N.S., Morris, D.T., Dew, P.M., and Pennington A., "An Evaluation of the Processor Farm Model for Visualising Constructive Solid Geometry," in *Parallel Processing for Computer Vision and Display,* Dew, P.M., Heywood, T.R., and Earnshaw, R.A., Eds., Addison Westley, pp. 443-451, 1989.

[Jain89] Jain, P. P. and Newton, P., "Putting Structure into Modeling," in *Proc. of the 1989 Summer Computer Simulation Conference,* Austin, Texas, USA, pp. 49-54, 1989.

[Jamieson87] Jamieson, L.H., "Characterizing Parallel Algorithms," in *The Characteristics of Parallel Algorithms*, Jamieson, L.H., Gannon, D., and Douglass, R.J., Eds., MIT Press, pp. 65-100, 1987.

[Nudd93] Nudd, G.R., Papaefstathiou, E., Papay, Y., et.al., "A Layered Approach to the Characterisation of Parallel Systems for Performance Prediction," in *Proc. Performance Evaluation of Parallel Systems (PEPS'93)*, Coventry, U.K., pp.26-34, 1993.

[Papaefstathiou93] Papaefstathiou, E. and Kerbyson, D.J., "Characterising Parallel Systems Focusing on Re-Usability," *PEPS Bulletin*, pp. 5-6, November 1993.

[Patel92] Patel, M. N., "Structuring Analytical Performance Models Using Mathematica," in *Proc. of the 6th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation,* Pooley, R. and Hillston, J., Eds., Ediburgh, U.K., pp. 273-286, 1992.

[Pease91] Pease, D., Ghafoor, A., Ahmad, I., Andrews, L., D., Foudil-Bey, K., and Karpinski, E. T., "PAWS: A Performance Evaluation Tool for Parallel Computing Systems," *IEEE Computer*, pp. 18-29, January 1991.

[Peps93] PEPS, *Characterising Processing Needs*, Tech. Rep. D5.1, ESPRIT Project 6942, University of Warwick, U.K., July 1993.

[Rolia92] Rolia, J.A., "Predicting the Performance of Software Systems," Ph.D. thesis, University of Toronto, 1992.

[Saavedra89] Saavedra-Barrera, R.H., Smith, A.J., and Miya, E., "Machine Characterization Based on an Abstract High-Level Language Machine," *IEEE Transactions on Computers*, vol. 38, no. 12, pp. 1659-1679, December 1989.

[Smith82] Smith, C.U. and Browne, J.C., "Performance Engineering of Software Systems: A Case Study," in *Proc. National Computer Conference,* vol. 15, pp. 217-224, 1982.

[Smith90] Smith, C.U., *Performance Engineering of Software Systems*, The SEI Series in Software Engineering. Addison-Wesley Publishing Co., Inc., 1990.

[Tron93] Tron, C. and Plateau B., "Modelling Communication in Parallel Machines within the ALPES Project," in *Proc. Performance Evaluation of Parallel Systems (PEPS'93)*, Coventry, U.K., pp. 110-117, 1993.

[Weishar87] Weishar, D.J., "Incorporating Expert Systems Technology into Software Performance Engineering," in *Proc. Computer Measurement Group,* pp. 720-722, 1987

[Zemerly93a] Zemerly, M.J., Papaefstathiou E., Atherton, T.J., Kerbyson, D.J., and Nudd, G.R., "Characterising Computational Kernels: A Case Study," in *Proc. Performance Evaluation of Parallel Systems (PEPS'93)*, Coventry, U.K., pp. 231-237, 1993.

[Zemerly93b] Zemerly, M.J., "Characterisation of Multi-Processor Systems," Research Report 256, University of Warwick, U.K., 1993.