In Proceedings of 19th IEEE International Performance, Computing, and Communications Conference (IPCCC 2000), Phoenix, USA, February 2000, pp. 485-492.

Performance Modeling of Parallel and Distributed Computing Using PACE

Junwei Cao[*] Darren J. Kerbyson[*] Efstathios Papaefstathiou[†] Graham R. Nudd[*]
[*]Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
[†]Microsoft Research Limited, Cambridge, CB2 4NH, UK
{junwei, djke, grn}@dcs.warwick.ac.uk efp@microsoft.com

Abstract

There is a wide range of performance models being developed for the performance evaluation of parallel and distributed systems. A performance modelling approach described in this paper is based on a layered framework of the PACE methodology. With an initial implementation system, the model described by a performance specification language, CHIP³S, can provide a capability for rapid calculation of relevant performance information without sacrificing accuracy of predictions. An example of the performance evaluation of an ASCI kernel application, Sweep3D, is used to illustrate the approach. The validation results on different parallel and distributed architectures with different problem sizes show a reasonable accuracy (approximately 12% error at most) can be obtained, allows cross-platform comparisons to be easily undertaken, and has a rapid evaluation time (typically less than 2s).

1. Introduction

Performance evaluation is an active area of interest especially within the parallel and distributed systems community where the principle aim is to demonstrate substantially increased performance over traditional sequential systems.

Computational GRIDs, composed of distributed and often heterogeneous computing resources, are becoming the platform-of-choice for many performance-challenged applications [3]. Proof-of-concept implementations have demonstrated that both GRIDs and clustered environments have the potential to provide great performance benefits to distributed applications. Thus, at the present time, performance analysis, evaluation and scheduling are essential in order for applications to achieve high performance in GRID environments.

The techniques and tools that are being developed for the performance evaluation of parallel and distributed computing systems are manifold, each having their own motivation and methodology. The main research projects currently in progress in this area include:

- *POEMS* [2]. The aim of this work is to create a problem-solving environment for end-to-end performance modelling of complex parallel and distributed systems. This spans application software, run-time and operating system software, and hardware architecture. The project supports evaluation of component functionality through the use of analytical models and discrete-event simulation at multiple levels of detail. The analytical models include deterministic task graph analysis, and LogP, LoPC models.
- *AppLeS* [10]. This is an application-level scheduler using expected performance as an aid. Performance predictions are generated from structural models, consisting of components that represent the performance activities of the application.
- *CHAOS* [12]. A part of this work is concerned with the performance prediction of large-scale data intensive applications on large-scale parallel machines. It includes a simulation-based framework to predict the performance of these applications on existing and future parallel machines.
- *Osculant* [13]. This is a class of bottom-up resource scheduler inherently suitable for heterogeneous information processing systems consisting of an arbitrary mix of processors, operating systems, application programs, and network topologies.
- *WARMStones* [1]. This work aims to provide a scheduler implementation toolkit, allowing researchers to implement their algorithms in isolation from dependencies on particular scheduling support system.

The motivation to develop a Performance Analysis and Characterization Environment (PACE) in the work presented here is to provide quantitative data concerning

the performance of sophisticated applications running on high performance systems [7]. The framework of PACE is a methodology based on a layered approach that separates out the software and hardware system components through the use of a parallelisation template. This is a modular approach that leads to readily reusable models, which can be interchanged for experimental analysis.

Each of the modules in PACE can be described at multiple levels of detail in a similar way to POEMS, thus providing a range of result accuracies but at varying costs in terms of prediction evaluation time. PACE is aimed to be used for pre-implementation analysis, such as design or code porting activities as well as for on-the-fly use in scheduling systems in similar manner to that of AppLeS.

The core component of PACE is a performance specification language, CHIP³S (Characterisation Instrumentation for Performance Prediction of Parallel Systems) [8]. CHIP³S provides a syntax that allows the description of the performance aspects of an application, and its parallelisation, to be expressed. This includes control flow information, resource usage information (e.g. number of operations), communication structures and mapping information for a parallel or distributed system.

In the work presented in this paper, the use of the PACE system is described through an example application kernel – Sweep3D [6]. Sweep3D is a part of the ASCI application suite, which has been used to evaluate advanced parallel architectures at Los Alamos National Laboratories. The capabilities for performance evaluation within PACE are illustrated through the cross-platform use of Sweep3D on both an SGI Origin2000 (a shared memory system), and a cluster of SunUltra1 workstations.

The rest of the paper is organised as follows: Section 2 describes the performance modelling approach based on the PACE conceptual framework. Section 3 gives an overview of the Sweep3D application and how it is described within CHIP³S performance specification language. Section 4 illustrates the performance predictions that can be produced by PACE on the two systems considered. Preliminary conclusions are discussed in Section 5.

2. PACE Performance Modelling Approach

The main concepts behind PACE include a layered framework, and the use of associative objects as a basis for representing system components. An initial implementation of PACE supports performance modelling of parallel and distributed applications from object definition, through to model creation, and result generation. These factors are described further below.

2.1. Layered Framework

Many existing techniques, particularly for the analysis of serial machines, use Software Performance Engineering (SPE) methodologies [11], to provide a representation of the whole system in terms of two modular components, namely a software execution model and a system model. However, for high performance computing systems, which involve concurrency and parallelism, the model must be enhanced. The layered framework is an extension of SPE for the characterisation of parallel and distributed systems. It supports the development of three types of models: software model, parallelisation model and system (hardware) model. It allows the separation of the software and hardware model by the addition of the intermediate parallelisation model.

The framework and layers can be used to represent entire systems, including: the application, parallelisation and hardware aspects, as illustrated in Figure 1.

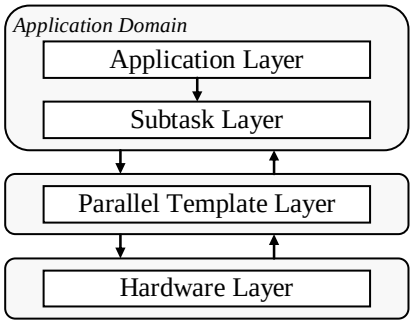

Figure 1. The Layered Framework

The functions of the layers are:

- *Application Layer* – describes the application in terms of a sequence of parallel kernels or subtasks. It acts as the entry point to the performance study, and includes an interface that can be used to modify parameters of a performance study.
- *Application Subtask Layer* – describes the sequential part of every subtask within an application that can be executed in parallel.
- *Parallel Template Layer* – describes the parallel characteristics of subtasks in terms of expected computation-communication interactions between processors.
- *Hardware Layer* – collects system specification parameters, micro-benchmark results, statistical models, analytical models, and heuristics to characterise the communication and computation abilities of a particular system.

According to the layered framework, a performance

model is built up from a number of separate objects. Each object is of one of the following types: application, subtask, parallel template, and hardware. A key feature of the object organization is the independent representation of computation, parallelisation, and hardware. This is possible due to strict object interaction rules.

All objects have a similar structure, and a hierarchical set of objects, representing the layers of the framework, is built up into the complete performance model. An example of a complete performance model, represented by a Hierarchical Layered Framework Diagram (HLFD), is shown in Figure 7.

2.2. Object Definition

Each software object (application, subtask, or parallel template) is comprised of an internal structure, options, and an interface that can be used by other objects to modify its behaviour. A schematic representation of a software object is shown in Figure 2.

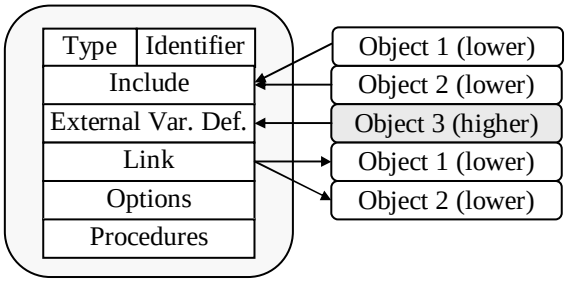

Figure 2. Software Object Structure

Each hardware object is subdivided into many smaller component hardware models, each describing the behaviour of individual parts of the hardware system. An example is shown in Figure 3 illustrating the main subdivision currently considered involving a distinction between computation, communication, memory and I/O models.

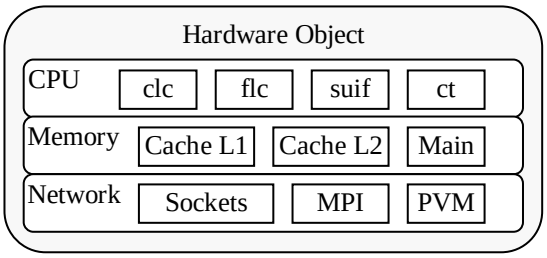

Figure 3. Hardware Object Structure

2.3. Model Creation

The creation of a software object in PACE system is achieved through the Application Characterization Tool (ACT). ACT aids the conversion of sequential or parallel source code into the CHIP³S language via the Stanford Intermediate Format (SUIF) [4]. ACT performs a static analysis of the code to produce the control flow of the application, operation counts in terms of high-level language operations [9], and also the communication structure. This process is illustrated in Figure 4.

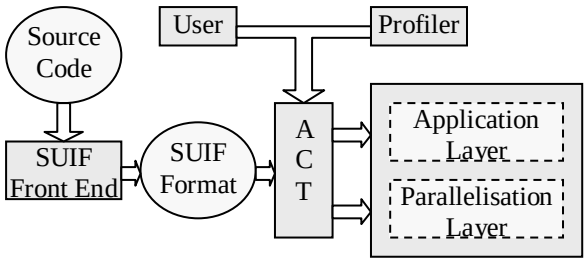

Figure 4. Model Creation Process with ACT

In PACE a Hardware Model Configuration Language (HMCL) allows users to create new hardware objects by specifying system-dependent parameters. On evaluation, the relevant sets of parameters are used, and supplied to the evaluation methods for each of the component models. An example HMCL fragment is illustrated later in Figure 9E.

2.4. Mapping Relations

There are strict mapping relations between source code of the application and its performance model. Figure 5 illustrates the way in which independent objects are abstracted directly from the source code and built up into a complete performance model which can be used to produce performance prediction results.

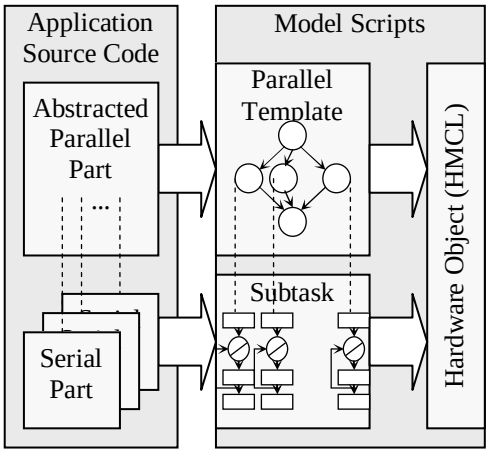

Figure 5. Mapping Relations

The mapping relations are controlled by the CHIP³S language compiler and the PACE evaluation engine, which will be described further in the next section through the use of the example application – Sweep3D.

3. Sweep3D: An Example Application

Here we illustrate the PACE modelling capabilities for performance prediction of Sweep3D - a complex benchmark for evaluating wavefront application techniques on high performance parallel and distributed architectures [6]. This benchmark is also being analysed by other performance prediction approaches including POEMS. This section contains a brief overview and the model description of this application. In Section 4 the model is validated with results on two high performance systems.

3.1. Overview of Sweep3D

The benchmark code Sweep3D represents the heart of a real Accelerated Strategic Computing Initiative (ASCI) application. It solves a 1-group time-independent discrete ordinates (Sn) 3D cartesian (XYZ) geometry neutron transport problem. The XYZ geometry is represented by a 3D rectangular grid of cells indexed as IJK. The angular dependence is handled by discrete angles with a spherical harmonics treatment for the scattering source. The solution involves two main steps:

- the streaming operator is solved by sweeps for each angle, and
- the scattering operator is solved iteratively.

A sweep (Sn) proceeds as follows. For one of eight given angles, each grid cell has 4 equations with 7 unknowns (6 faces plus 1 central); boundary conditions complete the system of equations. The solution is by a direct ordered solve known as a sweep from one corner of the data cube to the opposite corner. Three known inflows allow the cell centre to be solved producing three outflows. Each cell's solution then provides inflows to 3 adjoining cells (1 in each of the I, J, & K directions). This represents a wavefront evaluation in all 3 grid directions. For XYZ geometries, each octant of angles has a different sweep direction through the mesh, but all angles in a given octant sweep the same way.

Sweep3D exploits parallelism through the wavefront process. The data cube undergoes a decomposition so that a set of processors, indexed in a 2D array, hold part of the data in the I and J dimensions, and all of the data in the K dimension. The sweep processing consists of pipelining the data flow from each cube vertex in turn to its opposite vertex. It is possible for different sweeps to be in operation at the same time but on different processors.

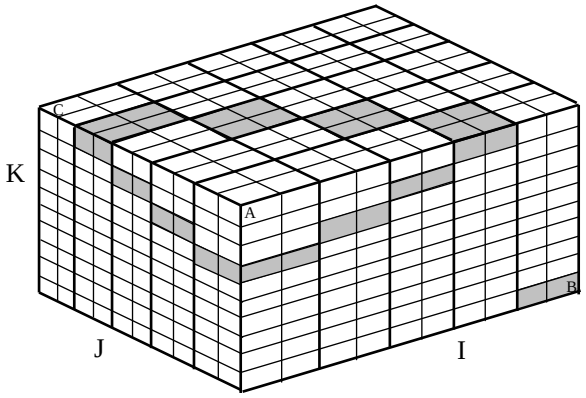

Figure 6. Data Decomposition of the Sweep3D Cube

For example, Figure 6 depicts a wavefront (shaded in Grey) that originated from the unseen vertex in the cube, and is about to finish at vertex A. At the same time, a further wavefront is starting at vertex B and will finish at vertex C. Note that the example shows the use of a 5x5 grid of processors, and in this case each processor holds a total of 2x2x10 data elements (data set of 10x10x10).

3.2. Model Description

We define the application object of the performance model as *sweep3d*, and divide each iteration of the application into four subtasks according to their different functions and different parallelisations. The object hierarchy is shown in Figure 7, each object is a separate rectangle and is labelled with the object name.

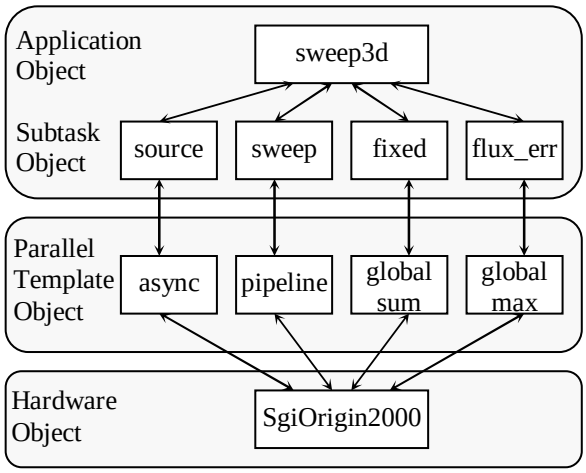

Figure 7. Sweep3D Object Hierarchy (HLFD Diagram)

The functions of each object are:

- *sweep3d* – the entry of the whole performance model. It initialises all parameters used in the model and calls the subtasks iteratively according to the convergence control parameter (*epsi*) as input by the user. Figure 8 describes different parts of the *sweep3d* object clearly in CHIP³S scripts. The sections correspond to those shown schematically in Figure 2.

```
application sweep3d {
    include hardware;
    include source;
    include sweep;
    include fixed;
    include flux_err;
    var numeric:
        npe_i = 2,
        npe_j = 3,
        mk = 10,
        mmi = 3,
        it_g = 50,
        jt_g = 50,
        kt = 50,
        epsi = -12,
        ......
    link {
    hardware:
        Nproc = npe_i * npe_j;
    source:
        it = it,
        ......
    sweep:
        it = it,
    ......
    }
    option {
        hrduse = "SgiOrigin2000";
    }
    proc exec init {
        ......
        for(i = 1;i <= -epsi;i = i + 1) {
            call source;
            call sweep;
            call fixed;
            call flux_err;
        }
    }
}
```

Figure 8. Sweep3D Application Object

- *source* – subtask for getting the source moments, which is actually a sequential process.
- *sweep* – subtask for sweeper, which is the core component of the application.
- *fixed* – subtask to compute the total flux fixup number during each iteration.
- *flux_err* – subtask to compute the maximum of relative flux error.
- *async* – a sequential "parallel" template.
- *pipeline* – parallel template specially made for the sweeper function.
- *globalsum* – parallel template which represents the parallel pattern for getting the sum value of a given parameter from all the processors.
- *globalmax* – parallel template which represents the parallel pattern for getting the maximum value of a given parameter from all the processors.
- *SgiOrigin2000* – contains all the hardware configurations for SGI Origin2000, which is comprised of smaller component hardware models already in existence within PACE. This can be interchanged with a hardware model of a different system, e.g. a cluster of SUN workstations.

The example model objects and their correspondence with the C source code is shown in Figure 9. Figure 9A is the C source code of showing part of the main function sweep, whose serial parts have been abstracted into a number of sub-functions in bold font. Figure 9C shows how the same source code structure is used to provide the parallel template description. Figure 9B is an example sub-function source code, which can be converted automatically to the control flow procedure in the subtask object as shown in Figure 9D.

Some of the main statements used in the CHIP³S language to represent the performance aspects of the source code are as follows:

- *compute* – a processing part of the application, its argument is a resource usage vector. This vector is evaluated through the hardware object.
- *loop* – the body of which includes a list of the control flow statements that will be repeated.
- *call* - used to execute another procedure.
- *case* – the body of which includes a list of expressions and corresponding control flow statements which might be evaluated.
- *step* – corresponds to the use of one of the hardware resources of the system. Its argument is used to configure the device specified in the current step. This is used in parallel templates only.
- *confdev* – configures a device. The meaning of its arguments depend on the device. For example, the device *mpirecv* (MPI receive communication operation) accepts three arguments: source processor ID, destination processor ID and message size.

It can be seen from the part of the Sweep3D model shown in Figure 9 that there is a lot of information extracted from the source code that is used for the performance prediction. The accuracy of the resulting model is of importance, and in Section 4 below, detailed results are shown to validate the model with measurements on the two systems considered.

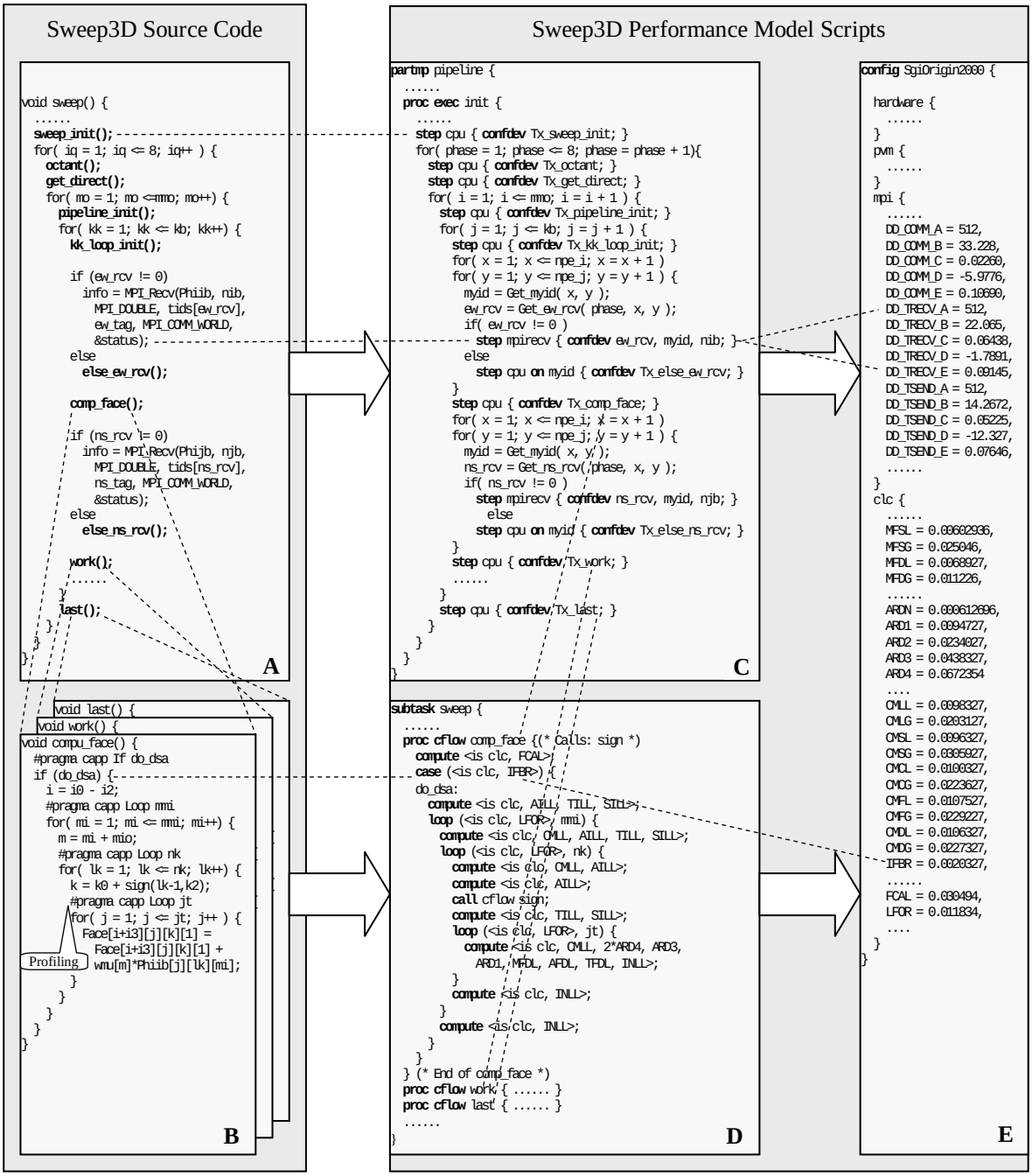

Figure 9. Mapping between Sweep3D Model Objects and C Source Code

Figure 9 also shows the inner mapping between the software objects and hardware object of the performance model. The abundant off-line configuration information included by the hardware object is the basis to implement a rapid evaluation time to produce the performance predictions.

4. Validation Results

In this section the preliminary validation results on execution time for Sweep3D are given to illustrate the accuracy of the PACE modelling capabilities for performance evaluation. The procedures in the PACE evaluation engine to achieve these results is complex and

out of the scope of this paper. Further details can be found in [7].

Figure 10 shows the validation of the PACE model against the code running on an SGI Origin2000 shared memory system. Note that the result for single processor input is not included because there are many special configurations, which are not included to current performance model for the sequential code. As shown in the figure, run time decreases when the number of processors increases. At the same time the parallel efficiency decreases too. In fact when the number of processors is more than 16, the run time does not improve any further.

By only changing the hardware object to the *SunUltra1* predictions on this new system can be obtained as shown in Figure 11. A cluster of 9 SunUltra1 workstations were used to obtain the measurements assuming no background loading. The run time spent is much more than that on SGI Origin2000 with the same workload. But the trend of the curve is almost the same.

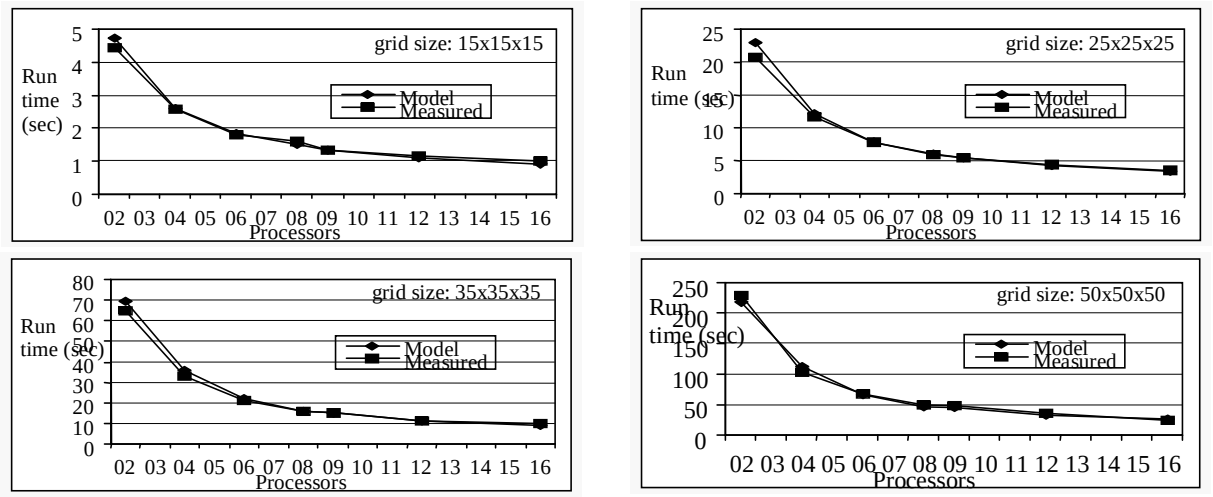

Figure 10. PACE Model Validation on SGI Origin2000

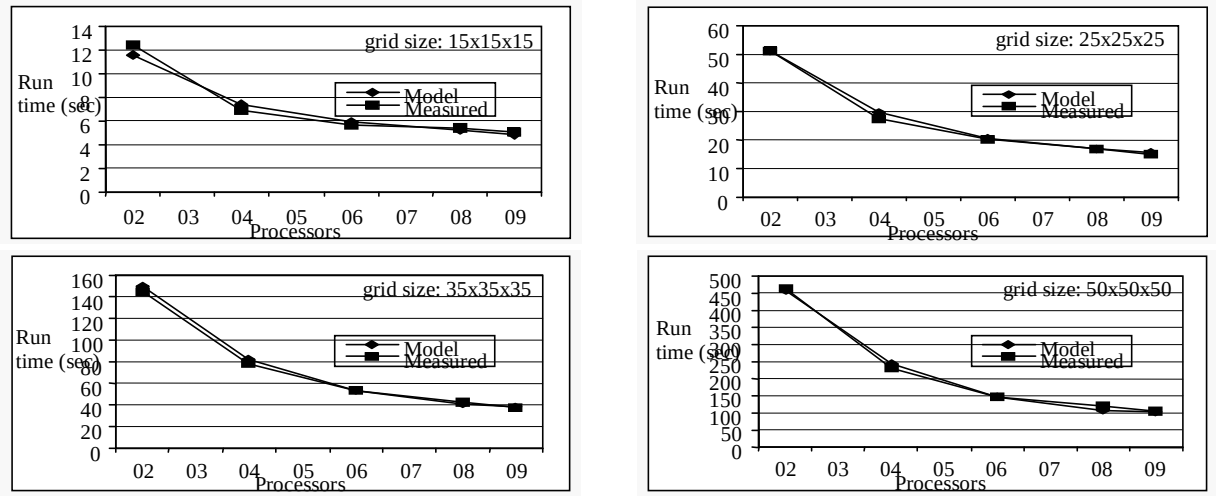

Figure 11. PACE Model Validation on Cluster of SunUltra1 Workstations

The accuracy of the prediction results were evaluated as follows:

$$\text{Error} = \frac{|\text{Measurement} - \text{Prediction}|}{\text{Measurement}} \times 100\%.$$

The errors between measurements and predictions are shown in Table 1, for the SGI Origin2000, and in Table 2 for the SunUltra1 Workstation Cluster. It can be seen that the maximum error is 11.44% in both cases, but the average error is approx. 5%.

Table 1. Prediction Error on SGI Origin2000

Err.(%)	15X15X15	25X25X25	35X35X35	50X50X50
1X2	6.53	10.44	7.02	-5.02
2X2	0.45	4.60	9.37	9.80
2X3	1.38	-0.73	4.47	-2.46
2X4	-5.66	0.82	1.12	-5.60
3X3	-0.29	-0.13	0.48	-4.55
3X4	-4.72	-4.92	-1.13	-7.62
4X4	-9.54	-4.90	-11.44	0.20

Table 2. Prediction Error on SunUltra1

Err.(%)	15X15X15	25X25X25	35X35X35	50X50X50
1X2	-6.79	0.15	3.24	-1.12
2X2	7.07	8.07	5.62	5.30
2X3	4.00	1.64	-0.20	0.32
2X4	2.85	-1.49	-4.30	-10.06
3X3	5.01	3.42	2.27	0.82

Besides the reasonable accuracy, the performance model can be used to obtain the evaluation results in a rapid time period, typically less than 2s. This is a key feature of PACE that enables the performance models to be used to aid to steer the application execution onto an available system at run-time in an efficient manner [5].

5. Conclusions

This work has described a performance modelling approach for parallel and distributed computing using the PACE toolset. A case study of the Sweep3D application has been given containing both model descriptions and validation results.

The key features of PACE include: a reasonable prediction accuracy – approximately 12% error at most; a rapid evaluation time – typically less than 2s for a given system and problem size; and easy performance comparison across different computational systems. It has been shown that the PACE performance system can produce reliable performance information which may be used for investigating application and system performance in many different ways.

The PACE system is currently being extended to provide support for performance prediction in computational environments which may be dynamically changing, and to aid the scheduling of multiple applications on the available resources. This corresponds in part to the challenges currently posed by the development of Computational GRIDs.

Acknowledgement

This work is funded in part by DARPA contract N66001-97-C-8530, awarded under the Performance Technology Initiative administered by NOSC.

References

[1] S. Chapin. WARMStones: Benchmarking Wide-Area Resource Management Schedulers. Draft white paper, http://www.cs.virginia.edu/~chapin/ws/ws.ps

[2] E. Deelman, A. Dube, A. Hoisie, Y. Luo, R.L. Oliver, D. Sundaram-Stukel, H. Wasserman, V.S. Adve, R. Bagrodia, J.C. Browne, E. Houstis, O. Lubeck, J. Rice, P.J. Teller, and M.K. Vernon, "POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational Systems", Proceedings of the 1st International Workshop on Software and Performance, pp. 18-30, 1998.

[3] I. Foster, and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufmann, 1998.

[4] M.W. Hall, J.M. Anderson, S.P. Amarasinghe, B.R. Murphy, S. Liao, E. Bugnion, and M.S. Lam, "Maximizing Multiprocessor Performance with the SUIF Compiler", IEEE Computer, Vol. 29(12), pp. 84-89, December 1996

[5] D.J. Kerbyson, E. Papaefstathiou, and G.R. Nudd, "Application Execution Steering Using On-the-fly Performance Prediction", in: High Performance Computing and Networking, Springer-Verlag, 1998.

[6] K.R. Koch, R.S. Baker, and R.E. Alcouffe, "Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor", Trans. of the Amer. Nuc. Soc., Vol. 65(108), 1992.

[7] G.R. Nudd, D.J. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper, and D.V. Wilcox, "PACE – A Toolset for the Performance Prediction of Parallel and Distributed Systems", to appear in High Performance Systems, Sage Science Press, 1999.

[8] E. Papaefstathiou, D.J. Kerbyson, G.R. Nudd, and T.J. Atherton, "An Overview of the CHIP³S Performance Prediction Toolset for Parallel Systems", in Proceedings 8th ISCA International Conference on Parallel and Distributed Computing Systems, pp. 527-533, 1995.

[9] B. Qin, H.A. Sholl, and R.A. Ammar, "Micro Time Cost Analysis of Parallel Computations", IEEE Transactions on Computers, Vol. 40(5), pp613-628, 1991.

[10] J.M. Schopf, "Structural Prediction Models for High-Performance Distributed Applications", Proceedings of 1997 Cluster Computing Conference, 1997.

[11] C.U. Smith, "Performance Engineering of Software Systems", Addison Wesley, 1990.

[12] M. Uysal, T.M. Kurc, A. Sussman, and J. Saltz, "A Performance Prediction Framework for Data Intensive Applications on Large Scale Parallel Machines", Proceedings of the 4th Workshop on Languages, Compilers and Run-time Systems for Scalable Computers, 1998.

[13] H. Wu, C. Chen, and F.J. Taylor. Osculant: A Multiprocessor Self-Organizing Task Scheduler. http://www.hsdal.ufl.edu/Projects/Osculant/osc06961.html