

Modelling Photochemical Pollution using Parallel and Distributed Computing Platforms

D. Abramson †

M. Cope §

R. McKenzie ‡

† School of Computing and Information
Technology

Griffith University

Kessels Rd, Brisbane,

Queensland, 4111

davida@cit.gu.edu.au

Phone: +61-7-875 5049

Fax: +61-7-875 5051

§ Victorian Environment Protection Authority

546 Collins St,

Melbourne, 3001

‡ Department of Computer Systems Engineering

Royal Melbourne Institute of Technology

P. O. Box 2476V,

Melbourne, 3001

Proceedings of PARLE-94, pp 478 - 489, Athens, Greece, July 1994

Abstract. Scientists are increasingly turning to numerical simulation in order to investigate and model complex environmental systems. Numerical simulation has enormous advantages over laboratory or field experimentation because it has the potential to allow a much greater parameter space to be considered. Numerical simulation is being used for simulating the formation of photochemical air pollution (smog) in industrialised cities. However, computational hardware demands can be great. In this paper we discuss the computational resources that are required in order to achieve a realistic number of results in a timely manner. We describe the parallelisation and distribution of programs that have been used as part of an air pollution study being conducted in Melbourne, Australia. We also describe some computer tools which proved effective in managing this study.

Keywords: Application of Parallel Systems, Scientific Computing, Design of Parallel Programs, Shared, Clustered and distributed machines.

1. Introduction

This paper concerns the parallelisation and distribution of some large scientific modelling programs which are used to compute the transport and production of photochemical smog within an urban airshed. The programs allow scientists in the Victorian Environment Protection Authority of Australia to simulate the smog production in Melbourne and to experiment with pollution reduction strategies. Simulation has major advantages over direct physical experimentation. It is possible to study many more scenarios than would be physically possible, and it is also possible to measure and assess the effect of control strategies without the enormous expense of implementing them.

Photochemical smog modelling poses some interesting challenges for computer architects. Such models consume enormous amounts of processor time and memory, and must often be performed within strict time constraints and budgets. Parallel and distributed computing technology has the potential to provide realistically priced platforms for performing such experiments.

In this paper we address two different techniques for providing the computational resource necessary to perform real air pollution studies. The system is used as part of the Melbourne Airshed study [1], and is currently being proposed for use in three other studies being performed in Australian cities.

The paper begins with a discussion of the techniques used in modelling photochemical smog, and then describes the computation structure of such calculations. We then describe the hardware infrastructure which was available, followed by the parallelisation and distribution techniques which were adopted for the Melbourne study. One of these activities required the construction of a tool to control the distributed experiments, and this is described in section 5.

2. Modelling Photochemical Pollution

One of the major uses of photochemical airshed models is to compute oxidant concentrations. Oxidants, such as ozone, are generated as a result of the chemical interaction between various precursors such as oxides of nitrogen (NO_x) and other reactive organics (ROGs) in the presence of ultra-violet radiation. Ozone is of particular importance because of its health related side effects; in Melbourne, Australia, the peak ozone levels have been observed to exceed 0.12 ppm in recent years, which is a widely adopted health standard level.

2.1 Prognostic Wind Field modelling

Meteorology is an essential component in the formation of photochemical smog. High temperatures and high incident fluxes of radiation are required to drive photochemical smog production, a process which typically takes 4-6 hours. Light winds and strong atmospheric stability are also prerequisites if high precursor and smog concentrations are to be generated. For a city such as Melbourne, these conditions exist during the summer months when synoptic winds are light. However, Melbourne's location, on a coast and surrounded by a bowl of hills to the north, east and west is such that complex meso-scale meteorological flows are usually present during these conditions. Successful simulation of photochemical smog production in turn requires the accurate simulation of these air flows.

Over the years, a number of prognostic mesoscale wind field models have been developed to compute 3-dimensional wind fields [2][3]. The model which was used in this work is based on the Colorado State University (CSU) mesoscale model, which has been shown to have good skill in predicting the wind fields where diabatic heating and cooling and orographic effects predominate. These conditions constitute a large proportion of oxidant event days within the Melbourne vicinity, making the model particularly suitable.

Wind field models attempt to simulate the processes important in the Earth's weather. They are based on the dynamics and thermodynamics of the large scale flow, as described by the Navier Stokes equations. Physical processes, such as the effect of solar and longwave radiation and interactions with the surface must also be included.

The partial differential equations that describe the atmosphere are highly non-linear and cannot be solved directly. Thus, some sort of discretization must be used to obtain

a numerically solvable system. There are many different mathematical schemes for solving such equations, however, these usually result in the solution of a number of finite difference equations. The various variables in the model are discretized over a finite grid, and the value of each variable at every grid point is computed. This computation is highly repetitive and lends itself well to vectorization and parallelisation, as will be discussed later in the paper. The modified form of the CSU code used in this work allows for multiple grid resolutions to be nested within each other, rather than applying one uniform grid to the region being modelled. This has the advantage that less work can be performed on regions of less interest to the study, however, it does complicate the parallelisation of the code.

2.2 Photochemical airshed modelling

There are a number of models capable of simulating the chemical transport and production that occurs in the atmosphere. During this work we experimented with two models. One was the Urban Airshed Model (UAM), which is a three dimensional Eulerian photochemical diffusion model developed by SAI [4]. The other is the CIT model, developed by McRae et al [5] originally at California Institute of Technology. Both perform the same basic computation, although they use different mechanisms for modelling the chemical interactions and different numerical procedures. Two distinct computations are performed. One is computing the concentrations of the various chemical species and the other is advection and diffusion of the species. The computation of the concentrations is performed by solving the following differential equation:

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (uc_i) = R_i.[c_1, c_2, c, \dots, c_n; T, t] + Q_i$$

where $c_i(x, t)$ is the concentration of species, $u(x, t)$ is the advective flow field, $R_i.[c_1, c_2, c, \dots, c_n; T, t]$ is the net rate of chemical production at position $u(x, y, z)$, temperature $T(x, t)$, and time t , and Q_i is the rate of direct emission of pollutant i above ground level. R_i is described by a set of coupled ordinary differential equations. This set of equations can be quite difficult to solve in practice because it is very stiff, and thus very small time steps are required. Consequently, quite a large amount of processor time is devoted to computing the chemical concentrations.

The rate of advection of chemical species i by the wind field may be written in two dimensional form as

$$\frac{\partial c_i}{\partial t} = - \frac{\partial uc_i}{\partial x} - \frac{\partial vc_i}{\partial y},$$

where u and v are the horizontal components of the three dimensional wind vector $U = (u, v, w)$.

In order to solve the equations given above, the three dimensional region is broken into a number of finite volumes and a finite difference form of the equations is solved numerically.

An important aspect of the computation of the species concentrations is that it is possible to compute the concentrations independently for each finite volume created

by the domain decomposition. However, information is conveyed between these volumes during diffusion and advection processes.

3. Computational Structure

The photochemical airshed models described in the previous section are used in two different computational experiments. One of these generates the ozone (and other pollutant) distribution for one particular emissions data base and weather scenario, and is used for event description and model verification. The other generates multiple scenarios and aggregates the results into one report. The latter is used to evaluate various control strategies.

3.1 Modelling a Single Scenario

One of the uses for the model is to test various control strategies and perform source sensitivity experiments. In this mode, the modeller interacts with the computer system by changing some important parameters and then performing a simulation. For realistic results a minimum, integration time of two days is required, and this can take up to 16 hours on a typical workstation. Such response times are far too long to be useful in performing source sensitivity experiments and severely limit the usefulness of the system. Typical control strategies may involve shutting down certain industrial sites for specified hours, reducing traffic flow along particular roads, etc. To achieve a turn around of about 1 minute, the model would require a sustained performance in the order of 3 Gflops which cannot be achieved without supercomputing hardware (either vector or parallel).

As part of the Melbourne study, a number of different visualisations were developed to display the results of the computation in a meaningful manner. Figure 1 shows the output screen of one of the visualisations of a single scenario. It shows the ground level concentrations of four different chemical species, Oxides of Nitrogen, Carbon Monoxide, Parafins and Ozone. The discretization of the grid into block shaped regions can be clearly seen. The display shows a false coloured image of the intensity of the species over the Melbourne region. The text on the screens corresponds to the suburbs in which air monitoring stations are installed. As the animation proceeds the pollutant concentrations can be seen to change and move.

This type of interface allows a modeller to experiment with different control strategies and get fast visual feedback on the effectiveness of the strategy. The mode of interaction clearly favours the application of lateral thought because the experiment is visualised in the language of the modeller.

3.2 Generating Ozone Contours - Multiple Scenarios

Another way in which the model is used is to determine the sensitivity of the photo chemistry to various input parameters. For example, it is common to plot the peak ozone concentration as a function of NO_x and ROG control. These diagrams then allow the modeller to determine the parameters that most affect the ozone level. The experiments are performed for a number of distinct physical locations. Unlike the single scenario case, these simulations do not require particularly fast response for any one run. However, the amount of work that is required to perform any one contour

chart is enormous; if a single scenario takes about 8 hours of workstation time, then one contour chart of 50 independent runs requires 400 hours to perform. If the charts are prepared for up to 10 different weather conditions and emissions data bases, then 4000 hours of compute time are required. Thus, even the contour charts require access to supercomputing technology in order to produce timely results.

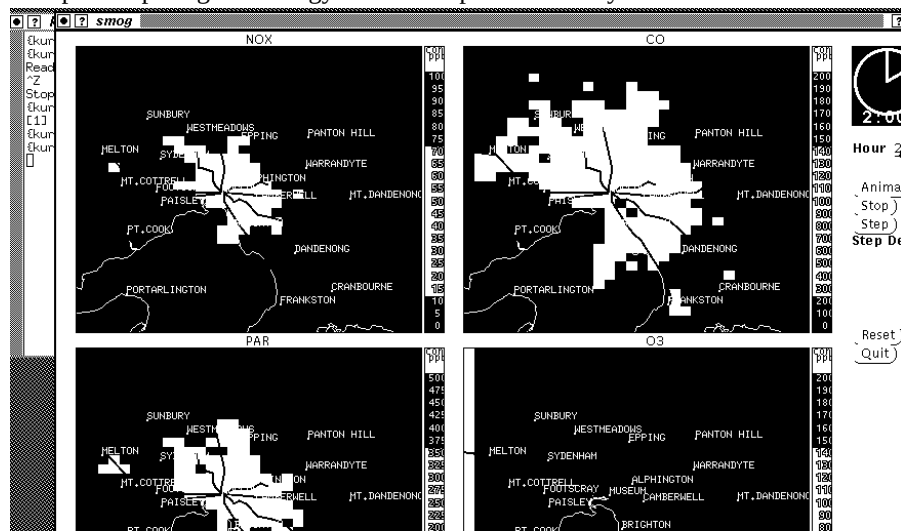


Figure 1 - Sample output from single scenario simulation

Figure 2 shows a sample ozone contour for one location in the Melbourne region. It clearly shows the non-linear effect of varying ROG_s and NO_x on the ozone concentration. In some regions of the control space, increasing one of the precursors can increase the ozone, and in others it can decrease the ozone. Thus, a simple minded strategy of decreasing the precursors may not have the desired effect of decreasing the oxidants. It is this effect that makes it essential to correctly model the process when evaluating control strategies.

4. Parallel and Distributed Platforms

4.1 Parallelisation or Distribution?

The two uses of the models described in the previous section lend themselves to two different parallelisation techniques and different hardware platforms. In order to generate one pollution scenario quickly it is necessary to deliver sustained super computer performance for the duration of the user interaction, ie. in the order of a minute. Clearly, the only way to deliver this level of performance is through either vector or parallel computing platforms (or a combination of the two). Consequently, the internal algorithms of the model must be vectorised or parallelised. Because one of our major concerns was cost we chose to parallelise the model and run it on a number of small parallel project level supercomputers like the Silicon Graphics Power Series [6]. Whilst this machine cannot deliver the sustained performed mentioned earlier in the paper, the study allowed us to evaluate the effectiveness of parallelising the model. There are now much larger and faster shared memory machines (like the CRAY T3D [7], the Convex SPP [8] and the Kendall Square Research KSR2 [9])

which are capable of delivering the required performance. These machine utilise a distributed virtual shared memory unlike the current shared memory machines.

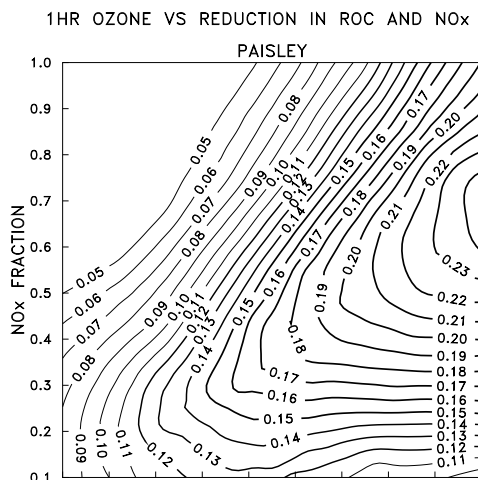


Figure 2 - Sample ozone contours

The generation of ozone contours requires massive computing capacity, but because of the way the output is used the information can be returned in the order of hours. This type of computation is well suited to a distributed platform because each model run is totally independent of every other. Thus, once the initial files have been generated, the runs can be distributed to workstations, and can consume excess unwanted machines cycles. In our case, we distributed the runs across many workstations at different physical sites. One problem with this approach is that the management of the distribution and collection of results can become complex, especially when workstations fail during the computation. To ease this problem we developed some specific tools to assist the process, and these are discussed later in the paper.

4.2 Parallelisation of Airshed Model

The primary aim of our work with the Urban Airshed Model (UAM) and the CalTech Model (CIT) was to determine the suitability of the central algorithms to parallelism on shared memory multiprocessors. Whilst we have parallelised both models, in this section we only present the results from the parallelisation of UAM. The computer used for this investigation was an 8 processor Silicon Graphics SGI 4D/380S.

Parallelising UAM began with identification of program regions which account for significant execution time. By examining time profiling results, the main users of CPU resources were found to be the vertical chemistry (STEP4 and subroutines), vertical advection (STEP3 and subroutines), and x/y integration code (STEP1/STEP2). With the vertical computations accounting for 69% of run time, STEP34 was the first to be parallelised, followed by STEP1 and STEP2 (10% of run time each). As the Power Fortran Analyser (PFA [10]) failed to parallelise the code effectively, the Silicon Graphics mp parallel directives were utilised to provide a

lower level of parallel programming. The main directive used in all cases was the DOACROSS statement which directs the compiler to automatically generate code to run iterations of a Fortran DO loop in parallel. A set of locks provided protection for a few dependent variables inside these DO loops.

The following table indicates the combined effectiveness of the above parallelisation procedures for a single 24 hr simulation. All times are averaged over 5 identical runs. From this it can be seen that a sequential simulation of a 24 hr scenario would have run for over 7 hrs. However with 8 CPUs available it will finish in just over 1 hr.

Threads	CPU secs	Speedup	Ideal	Efficiency (%)
1	25925	1.0	25925	100
2	13270	2.0	12962	98
3	9317	2.8	8642	93
4	7200	3.6	6481	90
5	5881	4.4	5185	88
6	5276	4.9	4321	82
7	4787	5.4	3704	77
8	4091	6.3	3241	79

Where:

CPU secs is the number of CPU seconds used by the master thread

Threads indicates the number of parallel processes chosen

Ideal is (CPU secs for 1 thread) / (Threads)

Speedup is the ratio (CPU secs) / (CPU secs for 1 thread)

Efficiency is (Speedup)/(Threads)*100

4.3. Parallelisation of meteorology

Parallelisation of the prognostic mesoscale windfield model proceeded in a similar manner to the photochemical airshed models. Again the computers used were Silicon Graphics Power series, initially a 4 processor machine and eventually an 8 processor machine. However, unlike the airshed models, in which a substantial proportion of computational time is spent in one section of the program ie the chemical kinetics solver, the computational time was fairly evenly spread across a number of routines. Therefore effective parallelisation of the model required that all routines be modified. Depending upon the structure of the source code, a mixture of coarse grain and fine grain parallelisation was adopted. For example routines which had minimal horizontal coupling such as the surface energy balance and surface temperature calculations, radiation calculations, pressure gradient calculations and vertical diffusion, could be parallelized at subroutine invocation level. On the other hand, horizontal advection, boundary condition routines and housekeeping routines had to be parallelised at the DO Loop level within the relevant routines.

The parallelisation of the code has proceeded in two phases. In the first sweep, an initial speed-up of 3.3 on the 4 processor machine was achieved and was deemed to be acceptable. However after installation on the 8 processor machine, a speed up of only 5.2 was achieved. Even though this indicates that more than 90% of the code was parallelised, this result was considered unacceptable and further optimisation was undertaken. In this second phase, attention was given to the parallelisation of smaller segments of code and also to restructuring the code in order to minimise cache misses.

This was achieved by reversing the indices of arrays for which operations only occurred in vertical columns ie SOIL(I,J,K) was replaced with SOIL(K,I,J) where SOIL contains the vertical temperature profile in the soil as a function of horizontal grid spacing and vertical diffusion is the only process considered. For cases in which arrays stored data which may undergo both horizontal and vertical processing ie horizontal advection and vertical diffusion of the vector wind field, vertical columns of the wind field were loaded into temporary one-dimensional arrays before commencement of the vertical processing. This was found to have a significant effect in cases where the vertical columns were accessed multiple times. A speedup of 6.8 was achieved as a result of these additional modifications.

Phase One				
Threads	CPU secs	Speedup	Ideal	Efficiency (%)
1	7610	1.0	7610	100
2	4013	1.9	3805	95
4	2279	3.3	1903	83
6	1688	4.5	1268	75
8	1471	5.2	951	65
Phase Two				
8	1125	6.8	951	85

Parallelisation of the model across multiple machines is also under investigation. As discussed earlier, the model is generally run as a series of nested grids of increasing resolution in which low resolution information from the outer grids is used to generate boundary conditions for the inner grids. Typically 2-4 levels of nesting are used. In initial attempts at parallelising this procedure have involved running the outer grid on a 4 processor machine, and an inner grid on the 8 processor machine. Boundary condition data for the inner grid is obtained via a shared disk drive. Initial attempts have been very successful however, full operational implementation of this scheme require the development of a dynamic scheduling procedure which ensures that optimum use of the processors are achieved. This becomes even more critical when more than two levels of nested are used and/or the machines are also loaded with other jobs.

4.4 Distribution of Airshed Model

The chemistry model reads input from a number of input files and emits concentration levels to output files. It reads a parameter file which contains the various file names and important parameters for the run. The model was sent to a large number of physically distributed machines, only some of which shared a file system. Thus, we could assume that files which were common across all runs could be simply placed in one shared position in the file system. Conversely, where machines did maintain a shared file system we wanted to take advantage of the fact and avoid duplication of data. In some cases of shared file systems, we were unable to use all of the machines connected to the system because there was insufficient space to create the output files.

Each time the model was exported for execution on another platform the files associated with the run were sent beforehand. We used the Berkeley Unix remote execution commands like *rcp* (remote copy) and *rsh* (remote shell) in order to control

the remote execution [11]. Free use of these commands required the addition of *.rhosts* files on each machine, which proved to be difficult because of security considerations. Consequently, on some machines we were forced to copy a *.rhosts* file to the target system immediately prior to remote job execution using *ftp* [12], and then to delete it after the run had completed. This was performed transparently by placing the commands in the queuing system scripts. Whilst this proved satisfactory it is a poor solution to the problem of accessing secure systems. For example, some secure systems refuse to acknowledge any remote execution commands even if there is a *.rhosts* file. This problem remains an open research issue for distributed computing environments.

5. A Tool for Controlling Distributed Computing

As discussed in the previous section we were able to gain significant computational resource by using workstations in our organisations. These machines, which were normally idle after hours, were used to process the multiple model runs required to generate the ozone contours. Since each contour diagram required in the order of 50 independent runs per chart, managing the process became quite difficult. Because of this, we developed a GUI based tool to control the generation, distribution and display of the results. There are a number of components to the tool.

First, it provides a mechanism for generating a series of model runs based on a set of parameters. In our case, we wanted to generate different simulations based on scaled versions of the emissions data base. Each run simulates a different scenario based on modified values for ROG and NO_x levels. Figure 3 shows a sample screen for controlling this generation phase. The user has the choice of selecting which model to use (UAM, CIT or another one called GRS [13]), which emissions data base to select and how to scale the ROG and NO_x levels. The levels are scaled by a fixed amount for a number of steps.

The second aspect of our tool is that it distributes the model across a number of workstations connected on the Internet. Figure 3 shows the tool configured to distribute across a number of local workstations; each box shows the status of a named machine. By clicking on a machine more information is produced about the system, and this is shown in Figure 4. The user can set the hours when the workstation is free to accept work. In this way, we were able to make use of machines after hours without disturbing the user of the machine during work hours. Information such as which job is currently being processed by the machine, what the load average is, etc. is also displayed in the host information screen.

The queuing system which underpins the tool distributes work until all of the simulations have been completed. If a machine becomes unavailable during processing it is removed from the set of machines for the remainder of the runs. After each job completes mail is sent to the user about the job. If anomalies have occurred during execution these are reported for user action. An additional load monitor was built so that if a user returned to a machine which was actively running a simulation, the simulation would be suspended until the machine became idle again. Using the queuing system we were able to make use of about 30 workstations, physically distributed between Melbourne and Canberra. Similar features are found in job

distribution systems like Condor [14], DQS [15], CODINE [16] and LoadLeveler [17].

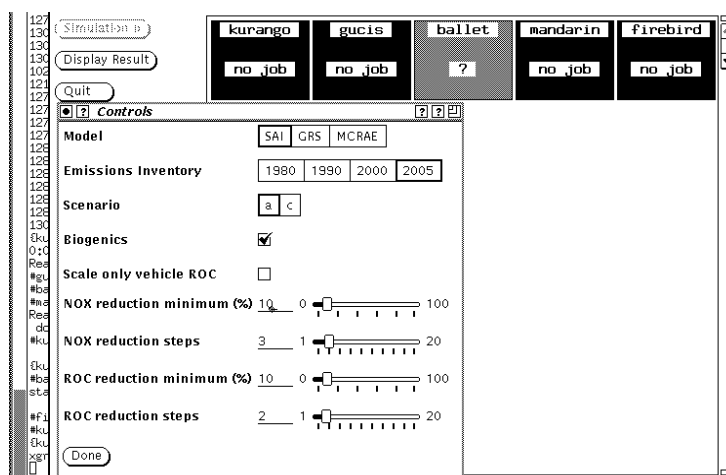


Figure 3 - Sample output of initial startup and status window

The third aspect of the tool is that when the experiment is complete the user can display the results in a condensed form. Figure 5 shows the results of a number of simulation runs. We have extracted one set of ozone readings for one physical site and plotted them against the scaled ROG and NOx levels. The results clearly show the effect of reducing either ROG or NOx on the ground level ozone concentration. Using this information the user may devise some ozone control strategies.

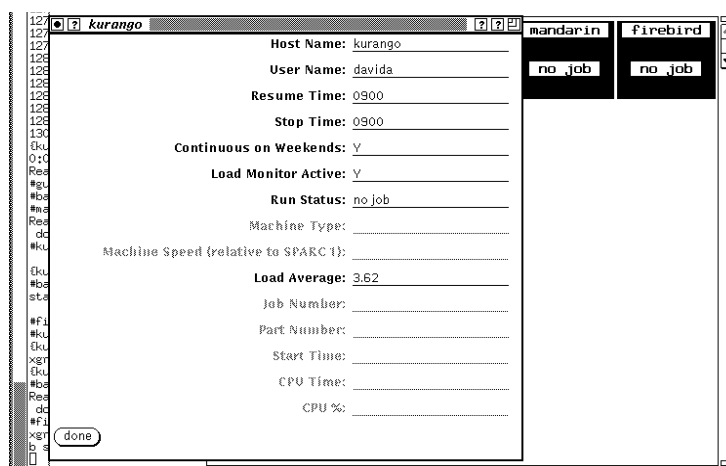


Figure 4 - Host information window

6. Conclusions and future directions

In this paper we have described two different methods for accelerated execution of photochemical smog simulations. One of these concerns the parallelisation of the

models for execution on parallel machines. The other involves the distribution of the model across many, often physically distributed, workstations. The latter form of execution gave us an enormous computational resource at no cost.

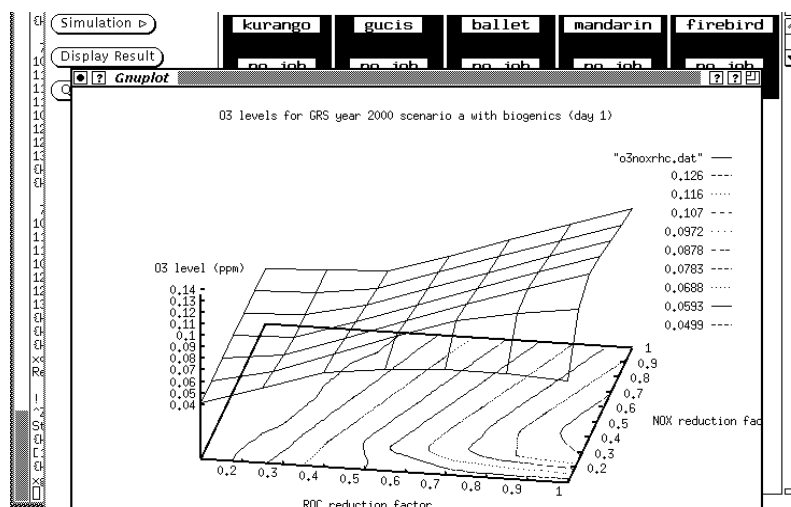


Figure 5 - Display of output from experiment

The parallel forms of the code were only developed for shared memory multiprocessors. This is because most of the available machines were shared memory. There are now a number of much larger, faster, multiprocessors in the Australian research community which rely on message passing code. Accordingly, we are currently parallelising a version of the CIT photochemical airshed model using message passing. Because the code was originally written in Fortran for sequential machines, this operation is proving more difficult than the shared memory port of the same code.

The tool for controlling the distribution was built specifically for controlling the photochemical pollution experiments. Similar parametric studies are also common in other areas of numerical simulation. For example, aeronautical engineers who study the performance of an aerofoil under different conditions may wish to generate a parameterised set of experiments and distribute them across a number of workstations. Consequently, we are developing a more generic tool for generating and controlling such parametric experiments. It will be applicable to arbitrary models which have been written without the knowledge that they will be distributed across multiple platforms. It will be built on top of the standard Distributed Computing Environment (DCE) to aid its use across open systems platforms. This work is being funded under the Co-operative Research Centre for Distributed Systems Technology.

Acknowledgments

This work was performed as part of a collaborative project between the Victorian Environment Protection Authority, the CSIRO Division of Information Technology and the RMIT Department of Computer Systems Engineering. Thanks go to the C.S.I.R.O. Division of Atmospheric Research for providing access to their 8 processor

Silicon Graphics machine on which some of these benchmarks were performed. Thanks go to Paul Pritchard and Binh Pham for reviewing a draft of this paper.

References

- [1] Cope M., Carnovale F. and Cook B., 1990. An Air quality Modelling System of Oxidant Transport and Production in Port Phillip Control Region. Proceedings of the 10th International Conference of the Clean Air Society of Australia and New Zealand.
- [2] Cope M.E. and Hess G.D., 1993. Meso-scale meteorological modelling for input into a 3-dimensional photochemical oxidant model. Proceedings International Congress on Modelling and Simulation, Perth, Australia.
- [3] Physic, W.L. 1988. Review: Mesoscale modelling in complex terrain. *Earth-Sci Rev.*, 25, 199-235
- [4] Morris R.E. and Myers T.C., 1990. Users Guide for the Urban Airshed Model: Volume I. EPA Research Triangle Park, N.C. USA (pub. EPA-450/4-90-007A).
- [5] McRae G.J., Russell A.G. and Harley R.A., 1992. CIT photochemical airshed model -users manuals, Carnegie Mellon University, Pittsburgh, PA and California Institute of Technology, Pasadena, CA.
- [6] Silicon Graphics, 1991. Power Series Performance Guide. Editor Veeleen Roufchaie, Member of Technical Staff Technical Marketing, ASD.
- [7] Cray Research, Inc., Cray T3D Technical Summary, September 1993.
- [8] Masson, Bob, "Bringing MPP to the Mainstream Customer", White Paper Series, Convex Computer Corporation.
- [9] Kendall Square Research, The KSR-1, Technical Summary.
- [10] Silicon Graphics, 1991. Power Fortran Accelerator Reference. Document: 007-0629-020.
- [11] Comer, D., Internetworking With TCP/IP, Volume 1, Second edition Prentice-Hall 1991
- [12] Postel, Jon, and Joyce Reynolds, "File Transfer Protocol (FTP)", Network Working Group Request For Comments - RFC 959, October 1985
- [13] Azzi M., Johnson G., Cope M., 1992. An introduction to the Generic Reaction Set photochemical mechanism. Brisbane: Proceedings of the 11th International Conference of the Clean Air Society of Australia and New Zealand.
- [14] Litzkow, M., Livny, M. and Mutka, M. Condor - A Hunter of Idle Workstations". Proceedings of the 8th International Conference on Distributed Computing Systems. San Jose, Calif. June 1988.
- [15] Green, T. P., Snyder, J. "DQS, A Distributed Queuing System", Florida State University, March 1993.

- [16] Ferstl, F, "CODINE Technical Overview", Genias, April 1993.
- [17] International Business Machines Corporation, "IBM LoadLeveler: User's Guide", Kingston, NY, March 1993.