

# Tool-based Parameterisation : An Application Perspective

**A. Lewis**

*Queensland Parallel Supercomputing Facility, Nathan, Qld 4111, Australia.*

**D. Abramson, R. Sosič**

*School of Computing and Information Technology, Griffith University, Nathan, Qld 4111, Australia.*

**J. Giddy**

*Co-operative Research Centre for Distributed Systems Technology, Nathan, Qld 4111, Australia.*

## 1. Introduction

As the new field of Computational Science develops a new approach to science is emerging. The collaborative research of mathematicians, computer scientists and scientists is resulting in an approach distinct from the traditional forms of theory and experiment. Increasingly, high performance computer technology is being used to analyse scientific problems, filling the gap between physical experiments and analytical approaches with numerical simulation. It is used to resolve problems intractable by conventional methods of enquiry, and allows new paths of research which could not be conceived otherwise.

Numerical simulation in itself is not new. Many large software codes have been developed for the simulation of a wide range of scientific and engineering problems. Mostly with primitive, file-based user interfaces, suited to batch execution and separate post-processing of results, they have been used successfully with vector supercomputers for many years. But now the growing availability of parallel supercomputers demands that new methods be found for their effective use, especially those that will protect the investment made in large, serial codes. Few generally applicable, readily scalable methods have been found. Often the use of parallel computers is disappointingly inefficient.

Computational science has spawned a large subset of cases in which supercomputers are used for parametric studies. In these studies a range of different simulations are calculated using the same code. One or more values are varied in the input data. For example, floating point or integer values may be varied uniformly between upper and lower bounds, or they may be given several discrete values, as when they act as seeds to stochastic processes. Text items may be given several values, perhaps defining different input files or even different versions of the simulation code to use. The cross-product of the varying items forms a set of possible combinations of input values. Each simulation run delivers results for one particular set of input values.

Using existing methods for such studies is extremely time-consuming and prone to error in the often complex parameterisation and job initialisation phases. Typically the user must manually create a set of input files for each of the parameter settings,

run the simulation program against each set of files, collate and merge the output files and produce some condensed form of output. Great care must be taken to maintain correlation between input and output.

In addition to these problems, each simulation run may take hours of execution time to produce one point in the output. When the effect on the output of several input variables must be explored, the combinatorial explosion can generate thousands of hours of serial computation. An efficient, easily applied method of using parallel supercomputers for these tasks becomes essential.

Until recently there have been two main ways of using parallel supercomputers for parametric experiments. One, through remote job execution of multiple jobs, and the other through distributed applications.

Several software packages exist to help manage queues of jobs for execution on remote computer systems. An example is the LoadLeveler product [1] used as the basic job distribution tool on the QPSF IBM SP2. Users generate a number of jobs and then submit them to the queue management system. The jobs are run on available processor nodes and the results returned to the controlling machine. An important advantage of this type of system is that the jobs are unaware of their distributed execution, so the application programmer need only generate a sequential task. The users, however, must concern themselves with generating and distributing multiple jobs.

Alternatively, distributed systems are built with the knowledge that they will execute on multiple processors. Many tools and methods exist for building distributed applications. For example, co-operating sequential processes may be used. A master process may automatically generate and distribute work across several processors, parameters being distributed through message passing. This approach has a major advantage over using a queue manager, because the user interface may be tailored to the application problem domain. It has the major disadvantage that the original code must be modified to manage distribution. This can require significant changes, especially as issues of fault tolerance become important with the long execution times that may be expected. In addition, programming for these environments generally requires a different skill-set to the original application programming.

## 2. NIMROD: A Tool for Parameterised Simulation

Nimrod is a tool that combines the advantages of remote queue management systems with those of distributed applications. It provides graphical user interfaces (GUIs) that give an application view of the experiment to the user and hides the details of job distribution and fault tolerance logging and restart. It covers three separate functions of job generation, job control and result aggregation.

**2.1. Job Generation.** The job generation phase is responsible for sending appropriate input files to the target processor via a remote file transfer server on the target system, and arranging execution via the job control mechanism.

An application user is presented with a customised GUI that allows specification of desired parameter settings within permissible ranges. Jobs are generated as a by-product of the different parameter settings. The cross-product of all legal parameter

values is generated by the system and each element of the resulting set is processed as an independent job.

**2.2. Job Control.** The job control phase is responsible for the execution of the individual tasks on the processor nodes of the target system. Jobs are distributed to computing resources concurrently by a queueing system. This is currently based upon an OSF DCE client [2] for dispatching jobs and remote execution servers for the individual processors. The DCE-specific information remains hidden from the application at the lower transport layers. It is possible to replace DCE with another transport mechanism, such as TCP/IP. Current development of the system involves, in part, investigation of the use of LoadLeveler for job control to streamline integration in the SP2 environment.

**2.3. Result Aggregation.** When all desired simulations are complete, the results must be presented in an appropriate manner. As this is highly dependent on the simulation task, Nimrod allows considerable flexibility by providing for execution of a user-supplied script to process and display the data in an arbitrary way.

Further details of Nimrod may be found in Abramson, *et al* [3].

### 3. Application Case Studies

Nimrod has been released for limited use and has to date been successfully applied to a range of engineering and scientific applications. In workshop sessions devoted to teaching the use of the tool several candidate applications have been considered. These workshop sessions have had the side-effect of demonstrating the ease with which the tool may be applied. In the space of a few hours, participants have acquired a thorough grasp of its workings, generated their own customised GUIs and in many cases initiated preliminary runs of the completed systems. The development of the GUIs is aided by use of a very simple, high-level description language tailored to the needs of the parameterisation tool.

A selection of example applications that have successfully been implemented using Nimrod are discussed below. A summary of the job characteristics may be found in Table 1.

**3.1. QED Modelling of Laser-atom interaction.** The study of the quantum electrodynamic modelling of laser-atom interaction is a typical example of a parameterised computational experiment. Atomic population dynamics are calculated for a particular atomic system for a range of laser intensities and laser detuning from resonance settings. In addition, the target isotope type, degree of Doppler broadening of the atomic beam and whether linearly or circularly polarised laser radiation is used to excite the atoms is chosen. Once the population dynamics are found it is possible to predict the Line Polarisation, or optical pumping parameter  $K'$  for circular polarisation, and investigate how they are related to the laser characteristics.

The application code was set up to sweep over ranges of laser detuning settings. It remained for the parameterisation tool to generate jobs for varying laser intensities. Each of these parameters were specified as floating point upper and lower bounds and desired number of points. From the combination of these two ranges, and for given

polarisation values, a three-dimensional surface of each of the optical parameters can be constructed. These surfaces were visualised and features interpreted. An example of such a visualisation is shown in Figure 1.

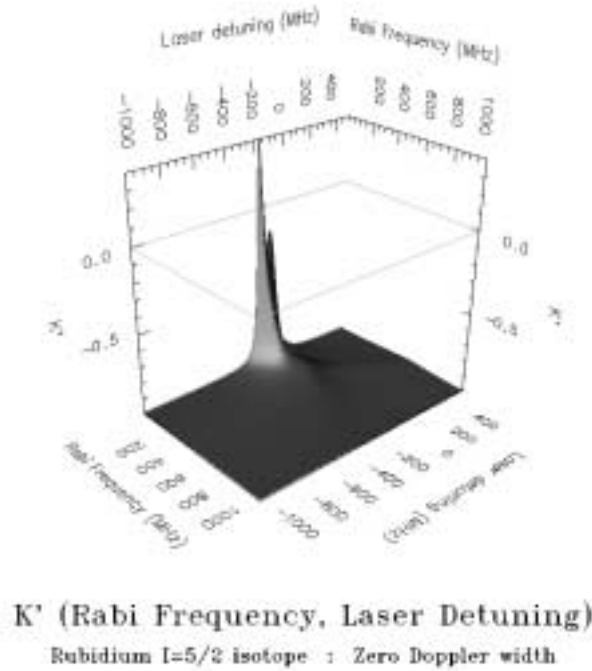


FIGURE 1. Example visualisation of QED modelling of laser-atom interaction

This example shows a previously undiscovered feature; the significant excursion away from the widely assumed constant value of  $K'$  for narrowly constrained values of laser detuning and intensity.  $K'$  is not easily experimentally measured. It is an important factor applied to results from physical experiments. As can now be seen, for lasers operating in certain regimes  $K'$  can vary considerably from its assumed value which may have profound consequences during interpretation of experimental results. Use of a parallel supercomputer with Nimrod is credited with making possible the investigation which discovered this feature.

Over the course of the investigation more than a thousand hours of cpu time were consumed. Use of Nimrod reduced the elapsed time for the critical experiments to a matter of a few days. This reduction was achieved by distributing the work over the nodes of the parallel supercomputer and allowing the application user to make easy and efficient use of the computer. The Nimrod overheads, essentially small file transfers at start and finish of each job, were negligible. Fully functional control scripts for Nimrod were developed in a few hours. If the user had been limited to the LoadLeveler queuing system, the parameter space searched would have been smaller and the resolution of the search coarser, because of the difficulties and time involved in coordinating and performing an exhaustive investigation. The programming effort

necessary to modify the code for truly distributed execution would also have been prohibitive, given the time constraints for the experiment.

**3.2. Modelling of Photochemical Pollution.** The modelling of photochemical pollution is another ideal application of Nimrod. The application is used to study pollution reduction strategies over a range of conditions without the enormous expense of implementing them in practice.

Photochemical airshed models are used 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<sub>x</sub>) and other reactive organic compounds (ROCs) in the presence of ultra-violet radiation. Ozone is of particular importance because of its health related side-effects. The time-varying, spatial distribution of pollutant concentrations are modelled for ranges of precursor concentrations under differing weather conditions.

Nimrod is used for modelling of the chemical reactions. The complex coupling effects consume enormous amounts of processor time making effective use of high performance computing resources essential. Nimrod proved an easy method to utilise parallel supercomputing and aided in the handling and processing of the considerable data produced. For a typical experiment there can be nearly a thousand input and output files, and nearly a thousand hours of serial computation. This produces results for just one set of weather conditions and emissions database. The overheads involved in transferring files to and from the remote nodes were negligible in comparison to the execution time of each simulation run.

**3.3. Trajectory Modelling for Atmospheric Tracers.** In conjunction with simulation efforts such as photochemical smog modelling, knowledge of likely concentration regions for released precursors can aid in effectively locating environmental sampling stations. For these purposes modelling tracer trajectories based on measured meteorological data forms an important pre-requisite study. While in computational terms these initial studies are of trivial extent, each run requiring only a few seconds of processing time, the possibility of error in job initiation is considerable. To produce each trajectory map, up to a hundred latitude–longitude pairs specifying tracer origins for multiple trajectories each day over periods ranging up to a year must be provided to the application. The parameterisation tool provided an error-free, user-friendly means of specifying these large input data fields.

This use of the supercomputer was not particularly efficient, the overhead in job initiation, file transfer etc. far outweighing the computation time. But the alternative of manual preparation, job execution and correlation of results is more time-consuming and so use of the tool considerably improves user productivity. The script used to generate the application GUI and control all phases of the Nimrod implementation is easy to construct. It is shown in Figure 2 together with the GUI produced.

**3.4. A Hybrid Approach to State Space Search.** Effectively reducing the steps of an iterative search and hence the search time of a given state space has until recently largely been dealt with by algorithmic methods. A hybrid method has been

TABLE 1. Summary of Job Characteristics.

	No. of jobs	Average Job Time	No. of Parameters and type
3.1	25–100	13 min.–2 hrs	$3 \times real, 3 \times boolean$
3.2	16–100	8 hrs	$2 \times real, 1 \times integer, 1 \times text$
3.3	100	1.4 sec.	$2 \times real$
3.4	25–100	30 sec.–10 min.	$2 \times integer, 1 \times text$

proposed that uses heuristic search to reduce the state space and branch and bound techniques to search the smaller resultant space.

As a basis for comparison, simulated annealing has been applied to standard problems. Several data sets were chosen and several random seeds used to initiate the heuristic search. Permutations of the search algorithm, for example by inclusion of pruning techniques, was also selectable. Each job returns a single number, the search time to minimise the function. These test systems may be executed repeatedly to assess the effectiveness of proposed improvements.

Nimrod allows rapid and easy setup of repeated large experiments and distribution of the tasks involved over the nodes of the parallel supercomputer. Its use is straightforward and the overheads minimal, consisting mainly of the distribution of the data sets.

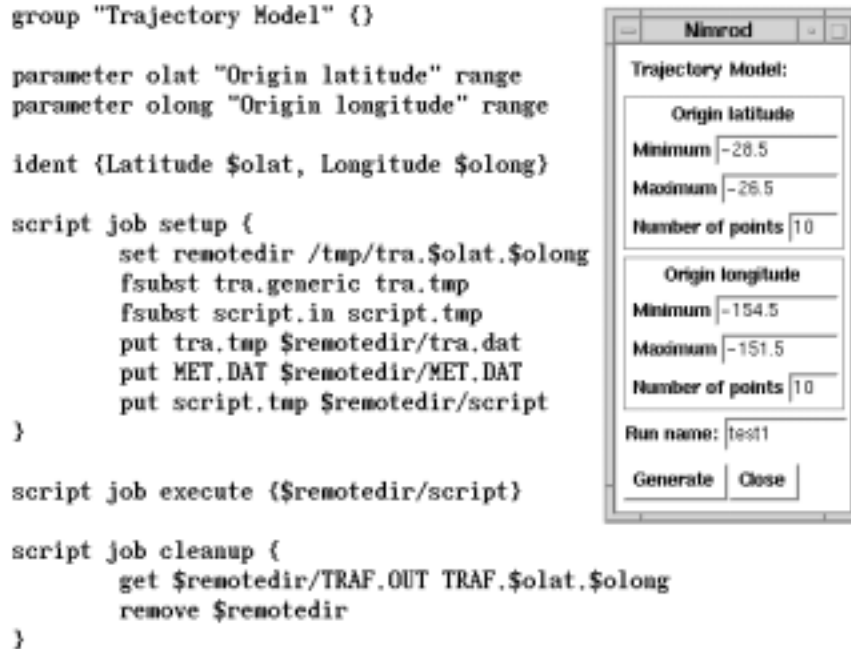


FIGURE 2. Example Nimrod script and GUI for Trajectory Mapping

#### 4. Conclusions

Scalable parallel supercomputers have the potential to deliver enormous computational resource. Computational science gives rise to an important class of tasks,

parameterised numerical simulation experiments, which can make very effective use of these resources. However, if use is to be made of these resources by specialists in applied disciplines, they must be presented in a manner relating to the problem to be solved. In this paper an extension to existing job distribution systems has been described that explicitly supports parameterised experiments.

Use of Nimrod has proved to be effective in providing efficient use of parallel resources to relatively unsophisticated end users. The improvement in execution times compared with serial execution on single nodes for computationally intensive jobs approached linear speedup. In addition, Nimrod delivered productivity increases in job initiation and work distribution, and a readily usable, user-friendly interface to high performance computing. This aspect of Nimrod made it the tool of choice even for users without large computational demands.

## 5. Acknowledgments

The tool described was developed at Griffith University as part of a collaborative project with the Co-operative Research Centre for Distributed Systems Technology (DSTC). The authors wish to acknowledge the support of Mr Melfyn Lloyd for proof reading a draft of this paper. Thanks go to our colleagues Brenton Hall, Suhail Khan and Marcus Randall from Griffith University, and Martin Cope from the Victorian Environment Protection Authority.

## References

- [1] International Business Machines Corporation, *IBM LoadLeveler: User's Guide*, 1994.
- [2] Rosenberry, W., Kenney, D. and Fisher, G., *Understanding DCE*, O'Reilly & Associates, 1992.
- [3] Abramson, D., Sosič, R., Giddy, J. and Hall, B., Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations, to appear in *Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995.