CODEMORE: A FAIR SYSTEM FOR DISTRIBUTING PARAMETER SEARCHES ON A CLUSTER COMPUTER SYSTEM

Paul HULSE, David DEWAR, Andrew COOPER

British Nuclear Group,

Risley, Warrington, Cheshire, UK.

and

Pat COWAN

Serco Assurance,

Winfrith Technology Centre, Dorchester, Dorset, UK

ABSTRACT

Criticality and radiation shielding considerations are two essential aspects of any safety case for a nuclear facility. In assessing these, it is usual to use a specialist computer code. The most accurate solution method is often Monte Carlo particle tracking, as used in the MONK and MCBEND criticality and shielding codes. It is now possible using CodeMore to do parametric searches of a model space using such codes on a cluster computer and to provide results with increased levels of confidence.

CodeMore has been developed as part of the Nuclear Codes Development (NCD) partnership between British Nuclear Group and Serco Assurance, to enable a fair distribution of parameter searches on a cluster computer system when using MONK and MCBEND. CodeMore extends the LANL WORM program so that it is compatible with the parameterisation syntax of MONK and MCBEND, and so that it interacts with a job scheduling system. CodeMore has been demonstrated successfully with both an in-house queuing system, and the OpenPBS scheduling system.

In this paper, we present the CodeMore program and show how it interacts with the job queuing system. We also show how CodeMore has been used operationally to significantly reduce the time needed to perform large classes of criticality problems, by automating the route to the solution and reducing the requirement for avoidable user input to the optimisation process. Importantly, we show how the availability of automated tools such as CodeMore, when combined with large cluster computer systems, have the potential to significantly alter the way in which shielding and criticality calculations are done, and to make possible calculation routes which were previously untenable.

1. INTRODUCTION

CodeMore has already been introduced as a method of distributing a Monte Carlo shielding calculation on a Beowulf cluster computer[1]. In general, CodeMore can distribute a parameterised criticality or dose assessment calculation over a cluster, significantly reducing the time taken to obtain a solution to complex questions in criticality safety or shielding analysis.

CodeMore extends the WORM program[2] to interact fairly with a job scheduling system. WORM was written primarily for MCNP[3], and it extends the input syntaax with embedded parameters and control structures. With a parameterised case, WORM parses the case file to produce a number of input files, one for each set of parameters. In this way WORM acts as a pre-processor such as `m4`[4] or `cpp`[5]. However WORM, and hence CodeMore, is in many ways more powerful then `m4` or `cpp` since it allows easy access to looping constructs and the use of fully functional Perl[6] subroutines. So Perl can be used, for example, to calculate atom number densities given a parametric material definition in an input file.

One of the design limitations of WORM is that it does not interact with a job scheduling system. WORM creates the input files from a parameterised job, however the analyst has to submit each job to a queuing system separately. Since most queuing systems operate on a First-In, First-Out (FIFO) basis, this can cause significant delays when used in a production environment, where there is a requirement for high priority jobs to override survey calculations. Although it is possible to allow for this by setting up multiple queues, it is more flexible if the parameterisation program takes care of all job submission and ensures that each parameterised job does not flood the queues. This is CodeMore's principal extension to WORM.

In addition, CodeMore internally converts the parameterisation syntax for the ANSWERS codes MONK and MCBEND to WORM syntax, so existing jobs using MONK/MCBEND parameterisation can be run using CodeMore.

2. CODEMORE

CodeMore uses WORM as its core code to handle parameterisation, but extends it to interact with various queuing systems. CodeMore is called with the parameterised input file as an argument. It then looks for a job control file that specifies what nuclear data libraries are to be hooked up and how the code is to be started. The CodeMore harness program creates a new executable, sets this running in the background and relinquishes control to the user.

The background program run by CodeMore monitors a single queue, checking for the number of currently scheduled jobs being below a high water mark. If this is true the next input file is immediately generated from the case file with the next set of parameters, and submitted to the queue. Between checking the queue and submitting jobs, CodeMore sleeps for a short period, with the sleep interval being randomly perturbed to prevent a first come takes all situation between competing CodeMores.

Multiple instances of CodeMore can effectively co-operate via the queuing mechanism, since CodeMore submits a single job each time it wakes, rather than immediately submitting jobs up to the high water mark. Using this mechanism, CodeMore can also co-operate with other users on the system, effectively getting out of the way if a high priority job arrives.

3. USING CODEMORE

Setting up a job to run using CodeMore is very straightforward. As an example, consider test case `ex1.nm` in the ANSWERS MONK 8b ru0 test suite. This models a stainless steel clad plutonium oxide rod, 52cm long by 8.5cm radius. The cladding has a 0.5cm curved wall thickness and 1cm end cap thickness. Consider the case where we wish to vary the length, L, between 40.0 and 152.0cm in steps of 1cm, and the radius, R, between 8.5 and 10.5cm in steps of 0.1cm. Figure 1 shows the necessary changes, in bold text, to the standard test case required to do this. Note that this uses WORM parameterisation, however MONK/MCBEND parameterisation could also be used.

```
BEGIN MATERIAL GEOMETRY
PART 1 NEST
ZROD M1
0.0 0.0 1.0
<R=8.5:10.5:0.1> <L=40:152:1>
ZROD M2
0.0 0.0 0.0
<R+0.5> <L+2.0>
END
```

Figure 1 - Parameterised MONK input file

Submitting a case parameterised as Figure 1 onto the BNFL Beowulf system (84 processors, each 2 GHz Intel Pentium 4 with 512MB RAM) using CodeMore creates a total of 2373 input/output pairs, with a total job run time of approximately 2 hours. The MONKCheck program [7], another code created as part of the NCD partnership, produces a summary table of the output files, which can then be processed in a number of ways. For example, it is possible to load this table into a spreadsheet and quickly produce a graph showing k-effective isopleths, as shown in Figure 2.

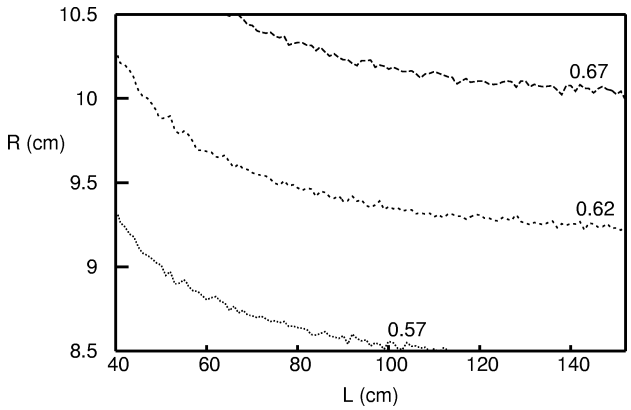

Figure 2 - CodeMore generated isopleths for the pin model

4. OPERATIONAL EXPERIENCES WITH CODEMORE

CodeMore significantly increases the capabilities offered to shielding and criticality modellers, especially when combined with a large computer cluster. As shown in section 3, CodeMore makes it simple to parameterise an existing model and do "what if" calculations. It is now routine, using CodeMore and a cluster, to run thousands of MONK cases for one analysis, either to find the optimal solution or to demonstrate that a particular solution is a worst case. The sheer volume of output data could become a problem in itself, without tools such as MONKCheck to handle the data flows and to provide a status check of the runs.

The remainder of this section looks at applications where the availability of CodeMore has radically changed how shielding and criticality calculations can be done.

K-effective Isopleth production

Before the introduction of high performance computers the production of k-effective isopleths was a slow and laborious process. In the past these were generated by choosing parameter values where the expected value of k would occur and then doing a small parametric study by hand. When coupled with a distributed computer system, CodeMore can quickly generate the base data for these isopleths, with minimum assessor involvement and with greater precision than before. Because of the flexibility of the WORM language used in CodeMore, it is also

possible to embed functions to perform complex tasks. For example, a function could be constructed to provide solution densities to give a constant mass of fissile material in a complex shape. Thus isopleths can be generated for systems which would have previously been thought untenable.

What if? modelling

Recent work within British Nuclear Group in the design of a new facility has very effectively used CodeMore as an integral part of What if? modelling for its design. Criticality safety has to be ensured, along with structural, heat and ventilation requirements. CodeMore was used to rapidly turn around a full criticality analysis as part of an interative design process. Potential solutions to this multiple criteria problem were found, and it was possible to work through to an acceptable design within a very short period.

Parallelisation of shielding calculations

CodeMore can be used to split a complex shielding calculation into a number of smaller calculations[1], and then distribute these cases onto a cluster computer. The results from the hundreds or thousands of output files can then be combined to produce a single result. Because these short cases use all the processors in a cluster, this method gives a solution much more quickly than on a uni-processor. Also, since each sub-job runs independently it is not necessary to dedicate part of a cluster computer to a single run, such as would be necessary with a PVM or MPI based solution.

5. FURTHER WORK

CodeMore is being further developed both to improve its portability and user image. CodeMore is developed in Perl, for which run time environments exist for various flavours of Unix, Microsoft Windows and Apple Macintosh OS. The principal portability issue is interfacing to different job scheduling systems. To date CodeMore can support an internally developed BNFL batch system and the OpenPBS system [8]. A principal future aim of this project is to extend the number of queuing systems supported, including for example Condor [9].

Improvements to the user image centre around possible future developments in fully integrating the CodeMore program within the Visual Model Editor system described elsewhere [10].

6. ACKNOWLEDGEMENTS

Thanks to Tom Jones, and the Los Alamos National Laboratory, for making the WORM code freely available for use in derivative works such as CodeMore. See `http://WORM.csirc.net/` for further details.

7. REFERENCES

[1] David Dewar, Paul Hulse, Andrew Cooper and Nigel Smith, “Efficient heterogeneous execution of Monte Carlo Shielding Calculations on a 'Beowulf' Cluster”, in 10th International Conference on Radiation Shielding (ICRS-10), and 13th Topical Meeting on Radiation Protection and Shielding (RPS-2004), Madeira, Portugal

[2] Tom Jones, “WORM (Write Once, Run Many) A General Purpose Input Deck Specification Language”, Los Alamos report 99-3594, 2000

[3] X-5 Monte Carlo Team, “MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume 1: Overview and Theory”, Los Alamos Report LA-UR-03-1987, April 24 2004

[4] Brian W. Kernighan and Dennis M. Ritchie, “The M4 Macro Processor”, in **Unix Seventh Edition Manual**, Volume 2, 1979

[5] GNU Project, **GNU C Preprocessor Manual**, Version 4.0.0, 2005

[6] Larry Wall, Tom Christiansen and Jon Orwant, **Programming Perl**, O’Reilly, 2000

[7] Keith Searson, “MONKCheck presentation”, The ANSWERS Seminar, 13 - 15 May 2003, `http://www.sercoassurance.com/answers/resource/areas/seminar/archive.htm` [checked 3rd May 2005]

[8] **OpenPBS Administration Guide**, `http://www.openpbs.org/docs.html` [checked 3rd May 2005]

[9] Condor Team - University of Wisconsin-Madison, **Condor Version 6.6.9 Manual**, `http://www.cs.wisc.edu/condor/manual/v6.6/` [checked 3rd May 2005]

[10] David Dewar, Andrew Cooper, Paul Hulse and Pat Cowan, “VME – A Visual Model Editor for the ANSWERS shielding and criticality codes”, in **Proceedings of the 11th International Conference on Information Systems Analysis and Synthesis: ISAS 2005 and The 2nd International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2005**, July 14-17, 2005, Orlando, Florida, USA