

Web Services for Remote Management of Scientific Simulations

K.A. Hawick and H.A. James
Institute of Information & Mathematical Sciences
Massey University – Albany
North Shore 102-904 Auckland, New Zealand
email: {k.a.hawick,h.a.james}@massey.ac.nz

5 March 2005

ABSTRACT

We describe a web services system for managing scientific simulation metadata. We show how the use of an appropriate data model based on sets can help define operations to specify; filter; reduce or expand against a set of simulation jobs, parameter values or even filenames. We describe a Java-based implementation of our ideas and discuss its implications for remotely managing and monitoring large scale scientific simulations on parallel and cluster compute systems in a grid environment.

KEY WORDS

service-oriented architectures; metadata; scientific simulation.

1 Introduction

Many areas of science and engineering make extensive use of computer simulation in the form of numerical experiments. Increasingly often this sort of work requires very large quantities of data [3] and the scheduling and allocation of expensive computer cycles [5]. This situation seems set to increase with simulation scientists needing to manage large data sets and often being limited in their endeavours by the complexities of the management task than by the inherent model formulation [6].

Increasingly this sort of work must take place at remote super-computational facilities or must be operated by a group of collaborating but geographically disparate colleagues. Web services [4] in support of numerical simulation experiment planning and operations are therefore very useful as they provide a common platform for collaboration.

We are ourselves working on large simulations in models in statistical physics and in the study of artificial life, but find similar issues constrain our colleagues

working computational chemistry and biology. Although much work has been done to advance the computational grid concepts for accessing and scheduling remote resources [9] we have found that managing the metadata for a set of simulations is itself a major task. Drawing from our past experiences in writing shell scripts and other ad hoc manual approaches to managing scientific simulation metadata [11], we consider in this paper the problem of providing a web services interface to an automatic management tool.

One motivating reason for creating this management tool was the need to monitor the progress of some extremely long-running simulations. We have been researching the behaviour of the “Small-World Ising model” under different lattice configurations [7]. In our study we have considered three sets of discrete parameters; in all the cross product makes for 780 discrete combinations. We need to run at least 100 jobs (with different random number sequences) for each of these 780 parameter combinations - making for a large operational management concern. Each data product from a set of 100 simulation runs is approximately 1GB, including the raw data and intermediate configuration state files. Producing each complete data product takes approximately nine processor days on our 130-processor “Helix” supercomputer cluster. We have found it useful (in fact absolutely critical) to develop and use a uniform naming scheme for files belonging to a given configuration. It is this naming scheme that we use to explain the features of our tool for remotely managing the compute jobs in section 3.

A second need for our tool arises because our cluster is shared between multiple (competing) users: the machine is not partitioned in such a way as to allocate blocks of nodes to one user, and it is not possible to reserve the dual-processor nodes to jobs that will only use one processor. This has often lead to the situation where one user’s job

consumes all the available disk space, leading to either some jobs not starting correctly, or worse, failing mid-execution and thus wasting machine cycles. Our experience has shown it is often more difficult to deal with those partially-completed jobs than it is simply to re-submit the same job at a later time. A Web services tool that allows remote monitoring of running jobs is useful to avoid wasting machine cycles and squeeze the most research value from a limited resource.

A third motivating reason is the need to work with colleagues and indeed amongst our own local team with the simulation management metadata. It is only of limited value to work with job parameters within a conventional spreadsheet. It is more useful to be able to interact with the actual data sets accumulated and planned on the supercomputers and to be able to link data file metadata dynamically into a Web services interface automatically. It is particularly valuable to be able to access status and planning information on our simulations from a mobile or remote location context WICAPUC [8].

This paper describes these ideas in the context of a prototype tool able to map metadata manipulations to actual data objects such as files or database entries on remote systems using Java-based [1] web services. We describe the underlying concepts and a notation for these operations in section 2. In section 3 we describe the specific web-services infrastructure we have built to experiment with these ideas. We discuss some general ideas on how such a system could be developed or customised for other simulation metadata management tasks in section 5.

2 Metadata Manipulation Concepts

The problem we are addressing can be described symbolically as follows. There is a set of data files \mathcal{F} that either already exist having been generated from some simulation jobs, or are in the process of being created. We want to inspect these files and the metadata that goes with them to help us collate the simulation data we have and need to meet the goals of our numerical experiment. Computational grid systems nearly always work entirely in terms of “files” [3] but it is advantageous to think of the data model in terms of a more generalised set of datum \mathcal{D} that could be files or could be large binary objects held in a relational database, locally or remotely. We consider the set \mathcal{I} of data identifiers as a set of “pure” names for our data. In practice a file name or database entry may not be pure- the filename or query needed to extract the data item is nearly always impure in that it has built in some meta data information that tells us something (extra) about the data referred to. Members of our set Our \mathcal{I} are treated as pure identifiers but will in practice have a one-to-one mapping with actual instantiations such as files or database entries.

We want to be able to remotely access a set of data instantiations and thereafter manipulate them as identifiers, being able to specify and invoke various operations on them. Operators (\widehat{O}_i) include those for Mutation (\widehat{M}), for Translation (\widehat{T}) and Expansion (\widehat{E}). Mutation operators will allow us to invoke commands (jobs) on datum. Translation operators allow us to move from different representations such as actual file names to a set of identifiers. Expansion operators will allow us to specify actual filenames to be used as targets for the data. Various status operators are also possible. Some simple example we have implemented are file size in bytes; file time-date stamp of type “timestamp”; file existence returning a logical true/false value.

Metadata field extraction and pattern matching operators \widehat{P} are easily specified using a pattern matching language such as the regular expressions library [12] found in many modern programming systems such as Java. We show how regular expressions operators can be packaged up as web service components in section 3 below.

Our goal then is to provide a web services infrastructure that allows a smart client program (such as a Java Applet or suitable set of plug-in script programs running at the client side of a web browser) to remotely gather, inspect and manipulate metadata pertaining to a set of remote partially complete files.

We have some set \mathcal{D} of data entities which may be ordered (in a sequence) or not. It may have been built up from different data in different locations. We want to map a particular data set \mathcal{D}_0 to a set of data Identifiers \mathcal{I}_0 in a one to one mapping. We then have possible operations on the set of identifiers to extract summary information from them using reduction operators, to re-expand them into data entities or mutation operators to signify we want some operation applied to the corresponding remote data entities.

We imagine operations like:

$$\mathcal{I}_0 = \widehat{T}(\mathcal{D}_0), \forall d \in \mathcal{D}_0, \forall i \in \mathcal{I}_0, i \leftrightarrow d \quad (1)$$

$$\mathcal{I}_1 = \mathcal{I}_0, \forall i \in \mathcal{I}_0, \widehat{M}(i) \quad (2)$$

$$\mathcal{I}_2 = \mathcal{I}_1, \forall i \in \mathcal{I}_1, \widehat{P}(i) \quad (3)$$

$$\mathcal{D}_1 = \widehat{E}(\mathcal{I}_2), \forall d \in \mathcal{D}_1, \forall i \in \mathcal{I}_2, i \leftrightarrow d \quad (4)$$

where we have a mutation operator:

$$\widehat{M}(i) = \text{Date}(i) > \text{Yesterday} \text{ AND } \text{Size}(i) < 100\text{KB} \quad (5)$$

and a pattern matching operator \widehat{P} which can be defined for some regular expression extraction and appropriate conditional on some extracted metadata field.

This describes the process of constructing the set of generalised identifiers for a set of data files; the selective filtering of recent files smaller than a certain size, the expansion of the identified set of data into filenames again.

Operations on \mathcal{D} (equations 1 and 4) take place at the server side, whereas operations involving \mathcal{I} can be conducted solely inside the client.

Other reduction operators that find the max/min of a set, or can report on the most recent n entries in a list are also possible and provide a means for a remote operator to inspect simulation run progress. We can therefore expect to conduct reasoning and plan operations on remote data while still at the meta data level. Operators can be combined and complex manipulations such as sorting, filtering by some criteria such as size; and combining by different metadata fields can all be carried out at the client end. Only when the final commit is made will complex command invocation be sent back to the supercomputer via the web server. Communications traffic is this minimised. We are trying to break up the phases so we can do as much lazy execution as possible and handle as much as possible at the metadata level.

3 Web Services Interface

In one sense what we want is more flexible version of the file browser dialogue found in Windows and Java and other component-based systems. We want to be able to invoke it remotely - with suitable user identification and verification. We want it to allow us to build up our data set using generalised identifiers and to be able to pick out extra information at the metadata level such as: file size; last modification date; and even existence. We want to be able to pull partial data from the filenames themselves (or database table names) and be able to use that metadata within our system to help us deduce information about the files or datum they contain without having to have an application-specific file reader program. We want all these ideas embodied in a web client with a supported set of web service components.

We would ideally like to have a generalised toolkit that can be customised for a particular simulation data management task. Also ideally the specification language for all these component services should be a standardised one supporting the maximal interoperability with other users' developed services. Our ideals are not quite attainable yet, but we show how a prototype can be realised for our Ising model simulation data example.

The shell script below shows how one might semi-automate a command on a compute cluster whose nodes are named in numerical order. Of course, this script relies on the presence of a shared home directory file system. We are not able to store data products on the shared home directories both due to the performance limitations of NFS (in the presence of large amounts of network traffic) and also space considerations. All data products are stored in per-node scratch-space directories.

```
#!/bin/sh
# usage: forall <file of node numbers>
#           <command to run>
for i in `cat $1`
do
    ssh helix$i $2
done
```

Scripts used to create, execute and clean up after jobs run on our cluster must take into account the fact that temporary scratch space needs to be used, and that space may already be consumed by another user's job (each of our cluster nodes has at least a dual processor which might be allocated to another user), so may need to find another free area of disk.

Typically, in order to check the progress of a simulation job running on our cluster, one must either perform a brute-force search across all the processor nodes to find an appropriately-named directory that corresponds to the job in question. We have introduced a further abstraction to help quickly locate running (or completed) jobs: when jobs are submitted, their details are added to a `jobrecord` text file, together with their PBS [10] job number. We have programmed the PBS output file for each job to print the node name on which the jobs is running. This file can be grepped for and the appropriate node found quickly.

Managing this suite of scripts is complex, but does not provide for easy access when one is not directly logged into the master cluster node. Packaging up the scripts as component operators in a web services framework adds considerably to their flexibility and usability remotely.

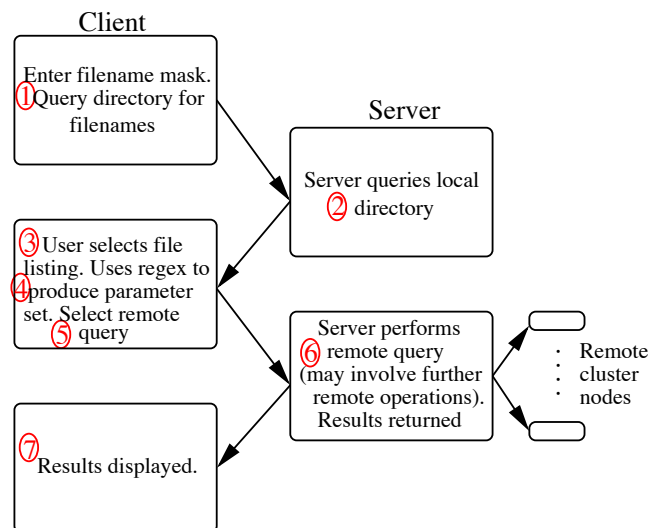


Figure 1: Sequence of transactions between web client and remote management server.

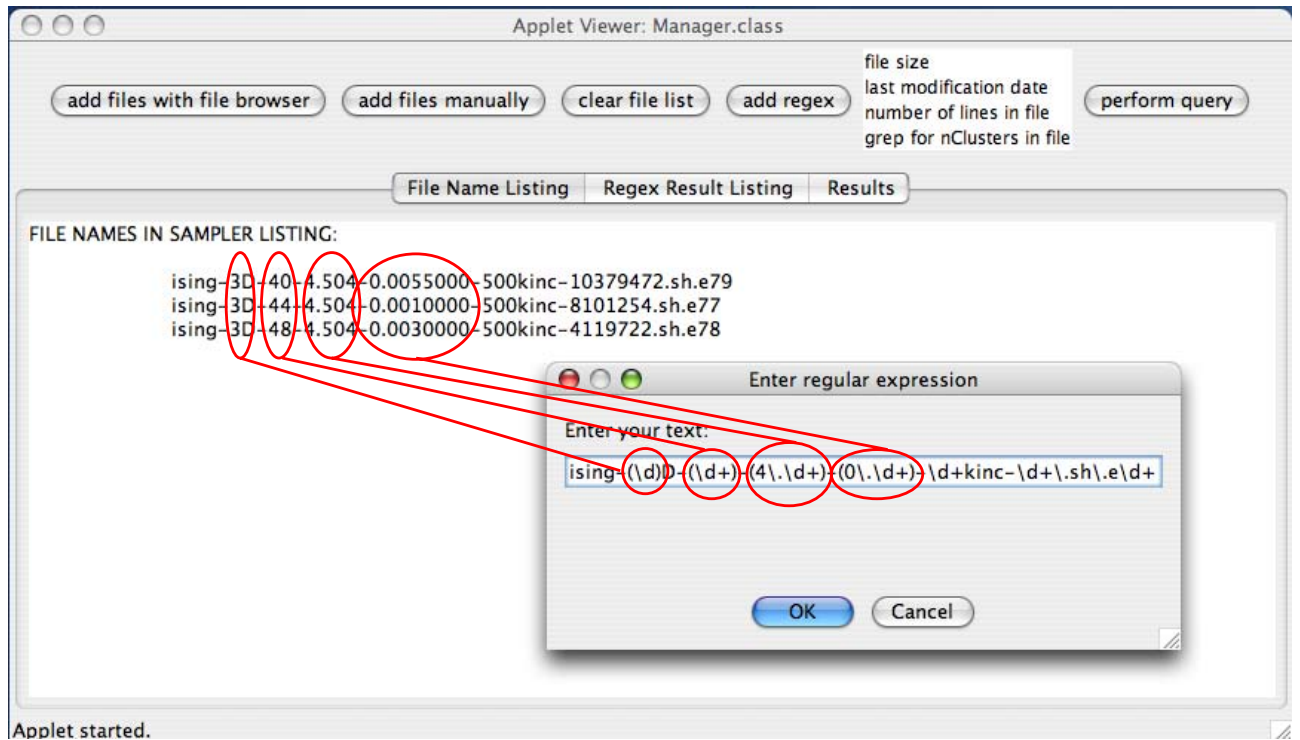


Figure 2: Specifying a regular expression allows the user to capture the variables they wish to use in posing a query to the remote server.

Figure 1 shows the sequence of transactions between the web client and remote management server. Users must specify a range of parameters with which they are going to query the remote server. In our experience the easiest way to do this is to use existing filenames containing the parameters of interest. We do note, however, that any suitable strings can be used. We omit the necessary (but important) initial steps of authenticating the user and providing a secure channel for communications. The steps are:

1. Due to Java's sandbox restrictions against browsing local filesystems we allow the user to query a directory local to the server.
2. The names of the files in the server's local directory are sent to the client.
3. The user can use standard techniques for selecting text from the file list.
4. The user specifies a regular expression (see figure 2) to parse the filenames.
5. The user selects what type of query they wish to run on the remote server. The system produces requests corresponding to the set of parameter cross-products (see figure 3).

6. The server performs the remote query, perhaps utilising further remote operations or web services.
7. The results are displayed for the user (see figure 3). Of course, the results can be directly used to specify additional file in step 3).

Figure 2 shows the operation of our prototype simulation manager. The user is able to enter (or swipe from another display, using the mouse) filenames that will act as prototypes for the remote data set they wish to investigate.

The verbatim text below shows the original file name of the scripts that are used to initiate the simulations under consideration. Each file name consists of: the dimensionality of the simulation, the lattice size of each dimension, the 'temperature' of the system, a probability with which the lattice is 'damaged', the number of iterations that the simulation will run for, and the random number seed that is used for this instance of the simulation.

```
ising-3D-44-4.504-0.0001000-500kinc-934392.sh
ising-3D-48-4.509-0.0055000-500kinc-4332.sh
ising-3D-40-4.513-0.0000300-500kinc-11427428.sh
```

In figure 2 the user has selected a menu option to enter a regular expression that will be used to parse the filenames, capturing and returning a set of parameters used to query

the remote server. Ellipses and lines have been added to the screen-dump to show how the regular expression capture groups correspond to the file names in the listing.

The verbatim text below shows the regular expression that is used to extract the pertinent details (dimensionality, lattice length, temperature and damage probability) from the three filenames above.

```
ising-(\d)D-(\d+)-(4\.\d+)-(0\.\d+)-\d+kinc-\d+\.sh
```

As we are using the Java implementation of regular expressions, the expressions in parenthesis are ‘captured’ for use outside the immediate regular expression parser. Thus, we can use the captured expressions later in the program: for each class of parameter (dimensionality, etc.) a set is created to hold all the discrete values of the parameter. The cross product of all parameter values is computed and is used as inputs to the appropriate remote query engine.

Figure 3 shows a superposition of two screen-shots from our system. The background screen-shot shows the cross-product of all the parameters captured by the regular expression in figure 2. In the foreground window the user has selected to test the last modification dates of the files with parameters matching the captured parameter cross-products. After the query has been performed and the server has responded, the results are displayed in the results window. The text reflects the fact that the tested files reside on a remote server (unavailable to the machine from which the Applet is run).

4 Discussion

We have restricted this paper to cover sets of data and identifiers with a one-to-one mapping. It is possible for our system to cope with ordered sets or sequences of datum. This is useful if there are dependencies of the processing of one item in a list with another. It is simpler to consider here however all sets to be unordered. We have also not discussed the issue of cached copies of datum. This is an important consideration in data grids as accessing network-nearby copy is obviously cheaper and faster than accessing a distant copy of a bulky item. Our system can cope with cached copies providing an extra effort is made to give each a unique identifier. This functionality can at least be separated out by a filter component concerned solely with these efficiency matters that exists between a client and server and acts during conversion from \mathcal{I} and \mathcal{D} .

Regular expressions have proved useful in constructing a repertoire of suitable web services to operate on metadata. These can be defined explicitly by an expert user at the client or as a set of canned operations available from

the menu. The canned options might be prepared an expert for use by other members of the simulation team as “black box” web services. Other application-specific programs could equally well be packaged up as operators. One problem that arises however is in managing large namespaces for all the operators.

We have simplified the description of the architecture of our system, avoiding reference to security issues, user credentials and access verification and authorisation. A smart client program can verify credentials with server security infrastructure separately from the main ideas we discuss in this paper.

Java is a good prototyping technology for these ideas. We were able to build a robust Applet client and custom server program quite rapidly. The prototype described in this paper represents less than a thousand lines of Java code, yet it has proved a worthwhile investment in terms of time saved in managing our simulations and also in terms of lowering the fraction of compute cycles lost due to complexity of operational management.

We anticipate that a more sophisticated program system could be built using some of the grid and web protocol packages becoming available in Java libraries. As with many previous middleware and metacomputing efforts however, we believe the long term value will be in the ideas and protocol standards rather than a short lived specific demonstrator program.

Along with many other computational scientists our traditional approach to managing simulations requires much hand editing of script files with cuts and pastes from directory listings. At the same time we often carry out calculations involving: total data sizes versus current disk capacity; estimated simulation completion times; and completion dates, either in an *ad hoc* manner (ie on paper) or within a traditional spreadsheet. The tool infrastructure we describe supports the concept of a “smart lab notebook” that integrates: job management; metadata inspection; experiment planning; and operational verification.

5 Summary and Conclusions

We have shown how a set of web services can be constructed around packaged operators to manipulate remote metadata pertaining to large scale numerical simulations. Tools such as Nimrod [2] and portal technologies [14] do allow remote job scheduling on clusters and supercomputer resources. We believe however that it is important to support an extensive collection of metadata manipulation services that allows more in depth parameter manipulation, selection and planning to be carried out at the web services client. Additional querying services for data dates, existence and size are also needed for a generic remote simulation management suite.

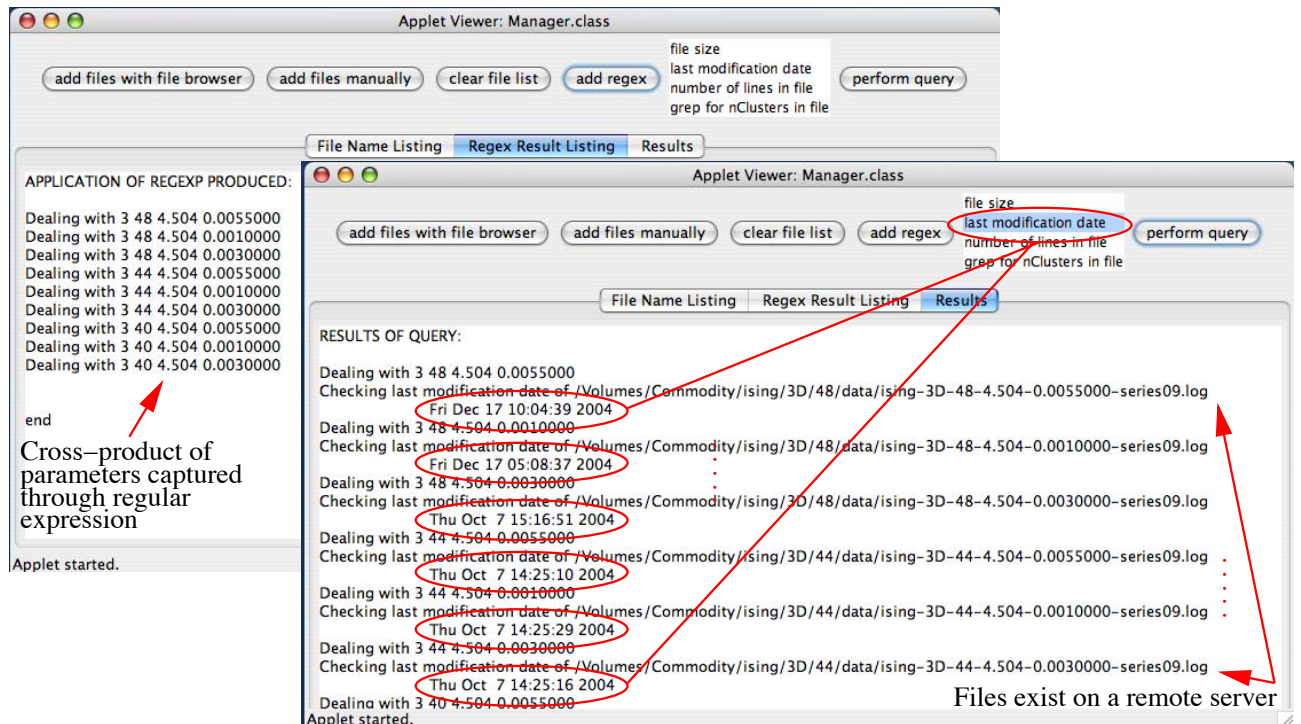


Figure 3: A superposition of two screen-shots from our system. In the background window the cross-product of the parameter values captured in figure 2 are shown. In the foreground window the results of performing a remote query are shown: the remote file names and their modification dates are shown. In this case individual files are kept on different machines in our remote cluster.

We have focused on the use of web services to manipulate scientific simulations, but these ideas likely extend to any situation where there is a need to manage bulk remote data where there is a complex set of associated metadata. We are investigating how some of the metadata may be held and manipulated in a form suitable for using semantic web services [13]. Standards are emerging for this and other computational grid protocols [15], but there appears still to be scope for much further work on interoperability standardisation of web services. We believe the general scientific data management issues described in [6] will need to be addressed with web-services tool development efforts like we describe.

Acknowledgements

The authors gratefully acknowledge the contribution of Helix supercomputer time by Massey University and the Allan Wilson Centre in the production of the data reported in this paper.

References

- [1] Sun Microsystems Inc. “Java Technology” Available from java.sun.com
- [2] D. Abramson, R. Sasic, J. Giddy, B. Hall. “Nimrod: a tool for performing parametrised simulations using distributed workstations,” In Proc Fourth IEEE Int. Symp. High Performance Distributed Computing - HPDC '95, IEEE Computer Society, 1995, ISBN:0-8186-7088-6.
- [3] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury and Steven Tuecke “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets” J. Network and Computer Applications, 2001
- [4] Francisco Curbera, William A. Nagy and Sanjiva Weerawarana “Web Services: Why and How,” Workshop on Object-Oriented Web Services, OOP-SLA, 2001
- [5] Dennis Gannon, *et al.* “Programming the Grid: Distributed Software Components, P2P and Grid Web

- Services for Scientific Applications,” *Cluster Computing* 5, pp 325–336, 2002.
- [6] Jim Gray, *et al.* “Scientific Data Management in the Coming Decade” in *CTWatch Quarterly*, February 2005. Available from www.ctwatch.org
- [7] K.A. Hawick and H.A. James. “Ising Model Scaling Behaviour on Small-World Networks,” Massey University technical note CSTN-006.
- [8] K.A.Hawick and H.A.James. “Middleware for Context Sensitive Mobile Applications,” in *Proc. Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive, and Ubiquitous Computing (WICA-PUC)*, Adelaide, South Australia, Feb 2003, as part of *Australasian Computer Science Week (ACSW 2003)*.
- [9] K.A.Hawick, H.A.James, A.J.Silis, D.A.Grove, K.E.Kerry, J.A.Mathew, P.D.Coddington, C.J.Patten, J.F.Hercus and F.A.Vaughan. “DISCWorld: An Environment for Service-Based Metacomputing” in *Future Generation Computer Systems* 15, 623 (1999).
- [10] R.L. Henderson and D. Tweten. *Portable Batch System Requirements Specification*. NAS Scientific Computing Branch, NASA Ames Research Center, California, April 1995.
- [11] H.A.James and K.A.Hawick. “Distributed Scientific Simulation Data Management,” Massey University Computational Science Technical Note CSTN-008, November 2004.
- [12] S.C. Kleene. ‘Representation of events in nerve nets and finite automata. In *Automata Studies*, Ann. Math. Studies No 34. Princeton Uni. Press, 1956, pp 3–41.
- [13] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. “Semantic Web Services”, *IEEE Intelligent Systems*, March/April 2001, pp 46–53.
- [14] Marlon Pierce, Choonhan Youn, Ozgur Balsoy, Geoffrey Fox, Steve Mock, and Kurt Mueller *Interoperable Web Services for Computational Portals*. SC02 November 2002
- [15] World Wide Web Consortium (W3C) Available from www.w3c.org