

Executing Large Parameter Sweep Applications on a Multi-VO Testbed

Shahaan Ayyub, David Abramson, Colin Enticott, Slavisa Garic, Jefferson Tan
*Faculty of Information Technology,
Monash University, Australia 3145*
{ayyub.shahaan, davida, cme, sgaric, jefferson.tan}@infotech.monash.edu.au

Abstract

Applications that span multiple virtual organizations (VOs) are of great interest to the eScience community. However, recent attempts to execute large-scale parameter sweep applications (PSAs) with the Nimrod/G tool have exposed problems in the areas of fault tolerance, data storage and trust management. In response, we have implemented a task-splitting approach, which breaks up large PSAs into a sequence of dependent subtasks, improving fault tolerance; provides a garbage collection technique, which deletes unnecessary data; and employs a trust delegation technique that facilitates flexible third party data transfers across different VOs.

1 Introduction

The computational Grid aggregates computational power and storage capacity by coupling together distributed CPU, network and storage resources [1]. The scale and nature of Grid testbeds make it possible to solve particular challenging problems in science and engineering using *parameter sweep applications* (PSAs). PSAs are generally specified as a set of experiments each representing a *parameterised instance* of a user-defined model. In principle, each instance is executed with a distinct set of parameter combinations on a number of resources. The parameters control the behaviour of the model and allow users to explore different design scenarios or options. PSA models have been applied in different areas including physical sciences [2, 3], mathematics [4] and biology [5]

Conceptually, implementing a PSA is not difficult. For example, a simple Cartesian product of a range of parameters would generate a set of experiments which can then be executed on a number of resources. However, in practice, effective deployment of PSAs on the Grid poses intricate management issues, making the handling of the experiments error-prone and unwieldy to perform manually. As a result, parametric modelling tools such as Nimrod [1], APST

[6] and GridBus [7] simplify this process. These tools allow e-scientists to concentrate on their science while hiding the complexities of deploying and managing applications on widely distributed resources.

Recently, we attempted to execute extremely large PSAs on the Grid, and this challenged solutions to task, data and trust management.

Task management: If an individual computation takes a long time, then executing it as one continuous run becomes error-prone and unwieldy. This is particularly true in the presence of unreliable network and resource conditions found on dynamic platforms such as the Grid. To address this problem, applications can be split and *check-pointed* [8]. In the event of a failure, a program can resume from the most recently saved checkpoint, rather than restarting the entire application. Whilst this idea is not new, we have implemented a simple user-driven technique supporting Grid-based check-pointing in order to achieve fault tolerance and improved application performance.

Data management: Task-splitting and check-pointing exacerbate data management problems because it becomes necessary to store application state for each checkpoint. Moreover, it might also be necessary to transfer the application's state to another resource. On the other hand, data transfer costs must be considered carefully. Scheduling heuristics such as [6, 9] are effective because they place downstream computations on resources such that the cumulative cost of transferring the associated files and estimated computation time is minimized. This improves the overall application performance by opportunistically migrating downstream sub-computations to better resources. However, the amount of data generated by multiple sub-computations may exceed the storage limits set on resources, which may restrict the further scheduling of jobs to those resources. To solve this problem, we built a garbage collection technique that keeps the amount of generated data within the permitted storage limits, even when multiple applications are simultaneously executed on a single resource.

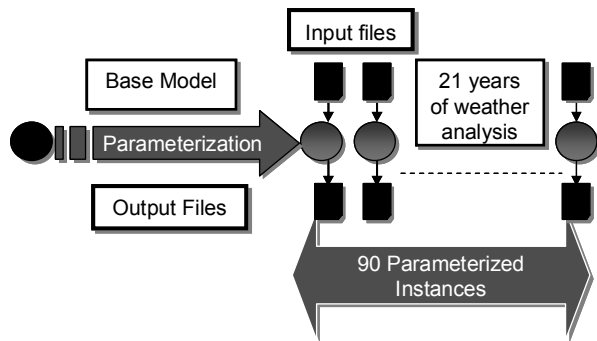


Figure 1. Old application model.

Trust management: Large-scale applications require as many computational resources as possible to achieve high throughput. Unfortunately, a single *virtual organization* (VO) [1] may not provide sufficient resources, so it may be necessary for a Grid testbed to span multiple VOs. Typically, Grid resources belonging to a particular VO only trust users authorized by the *Certificate Authority* (CA) of that VO. As a result, data transfers across resources (often called a *third party* transfer) belonging to different VOs may not be allowed. To address this, we have implemented a trust delegation model that facilitates flexible third party data transfers across different VOs.

The devised techniques are implemented in Nimrod/G, a tool which executes Grid-based PSAs [1]. As a result, Nimrod/G gracefully migrates the downstream sub-computations to better resources when a job or resource fails or when there is resource contention. Moreover, Nimrod/G efficiently co-schedules multiple computations on a single resource, while keeping the cumulative data generated by these computations within permitted storage limits. Further results show that whilst migrating sub-computations to resources of different VOs, Nimrod/G appropriately copies the state files associated with every sub-computation.

The next section describes challenges encountered whilst executing large PSAs on a multi-VO Grid. Section 3 describes our implemented techniques and the resulting changes to the application model. Section 4 describes the implementation of our techniques in Nimrod/G. Section 5 illustrates the efficacy of the implementation through experimental analysis. Section 6 concludes the paper and mentions some of our future endeavours.

2 Challenges

The experiment described in this paper studies the effects of the burning of the savannah on the climate of Northern Australia [10], using the Conformal-Cubic

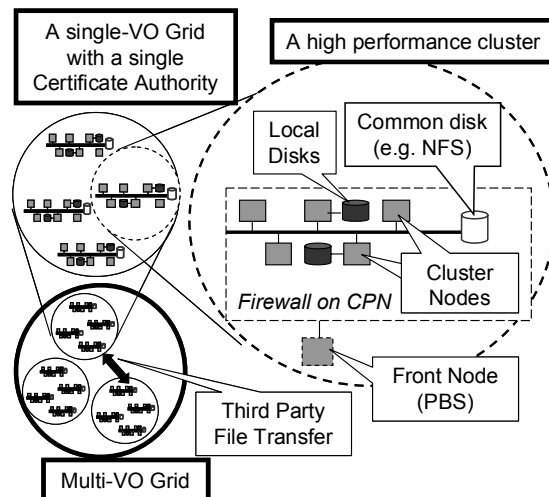


Figure 2. Grid testbed model.

Atmospheric Model (CCAM) [11]. The inputs to CCAM, which includes a set of files, can be tuned to simulate different climatic scenarios of a particular geographic location. For example, one of the parameters, the time- parameter, specifies the duration for which the analysis should be performed on a particular geographical location. We generated an order of 90 instances of CCAM for the experiment [12].

2.1 Task Management

The application was initially set up as shown in Figure 1. Each instance takes a file as input, computes a scenario and then generates an output for further investigation. Each computation modeled 21 years of weather conditions requiring 6-7 weeks of CPU time to complete. As mentioned earlier, executing such long running jobs as a single-continuous run is prone to failures. Furthermore, restarting a long running computation (6-7 weeks in our case) due to a job or resource failure would waste huge amounts of time already invested in the failed run and additional time to re-run the computation from the very beginning.

2.2 Data Management

Task-splitting, as described earlier, generally adds to the data management problem. On the other hand, appropriate task allocation techniques can improve an application's performance by migrating the downstream computations to resources with better CPU performance and network connectivity. However, existing heuristics assume unlimited disk space on Grid resources [13, 14], which is unrealistic on Grid testbeds where users are typically allocated

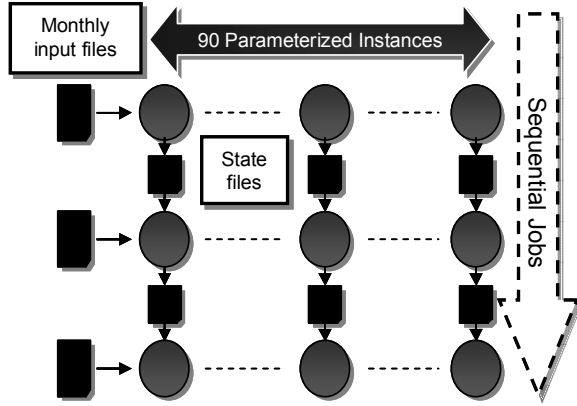


Figure 3. New application model with sequential dependencies.

limited disk space. In our test case, each sub-computation generated about 70MB of output. Combined with the common input files of around 250MB in size, this exceeded the storage available on some testbed resources. On the other hand, much of the output data is only required temporarily since the output of one sub-task is required only as input to the next sequential task.

2.3 Trust management

To deal with the scale of our application, we executed our experiments on the Pacific Rim Applications and Grid Middleware Assembly or PRAGMA Grid testbed [10]. The PRAGMA testbed consists of a large set of resources spanning across several VOs. The Globus toolkit [1] was used to build the basic infrastructure for providing resource access and remote data transfer services.

Figure 2 shows a sample of such Grid testbed models integrated from several individual VOs. Typically, all resources in an individual VO trust a single certificate authority (CA) [1], and thus all resources can communicate freely and authenticate data transfers against this single CA. This is important when intermediate files are transferred directly from one resource to another as part of a data-dependent computation within the PSA. However, if the testbed spans multiple VOs as discussed, then a sending resource and a receiving resource may trust different CAs, making third party transfers difficult. One solution to this problem is to route the data through a resource trusted by both parties. However, this incurs additional communication overhead, as the data now needs to be transferred through an intermediary resource. To address this, we have designed and implemented a new mechanism that delegates trust across multiple VOs, allowing the direct transfer of data between resources in different VOs.

```

1. parameter exprnr integer range from 1 to 90 step 1;
2. seqameter timestep integer range from 1 to 252 step 1;
3. task main
3.1. inputs cms.${exprnr}.`/bin/echo ${$(timestep) - 1}`.tar.gz
    vf_ccam.$timestep
3.2. output cms.${exprnr}.$(timestep).tar.gz on create fl
3.3. sequmeters timestep
3.4. parameters exprnr
3.5. node:execute tar xfz cms.${exprnr}*.tar.gz
3.6. node:execute /bin/mv vf_ccam.$timestep
    cms.$exprnr/input/vf_ccam.$timestep
3.7. node:execute cd cms.${exprnr} && /cms_prep.ksh
    ${exprnr}$(timestep) >> ../../output 2->1 && cd ..
3.8. node:execute cd cms.${exprnr}/output && .
    ${HOME}/bin/jobpea >> ../../output 2->1 && cd ../../
3.9. node:execute cd cms.${exprnr} && /cms_postp.ksh
    ${exprnr}$(timestep) >> ../../output 2->1 && cd ..
3.10. node:execute tar cfz cms.${exprnr}.$(timestep).tar.gz
    cms.${exprnr}
4. endtask
5. task fl
5.1. parameters None
5.2. seqameters None
5.3. copy node:$FILE local,root:
5.4. node:execute $HOME/$replicate s brecca-2.vpac.org Obj1
6. endtask

```

Figure 4. New planfile syntax.

3 Solutions

3.1 Task Management

Many scientific applications, such as CCAM, provide explicit check-pointing at the application level. In our case, we split large simulations, each computing 21 years of weather scenarios, into a sequence of 252¹ dependent monthly sub-computations, linked through *state files* containing check-pointed data (Figure 3). A state file is an intermediary file containing the state information from one subtask that is to be used by its subsequent child.

Dependencies resulting from task-splitting, such as addressed in this paper, are difficult to express succinctly in existing parametric modeling languages embedded in existing tools such as GridBus [7], ICENI [15] and earlier versions of Nimrod/G [1]. Other declarative languages, such as APST's [16], use comprehensive but large XML-based descriptions to specify experiments [17].

Nimrod/G uses a simple declarative language, called *planfile*, to specify parametric modeling experiments. Figure 4 shows the syntax of the current implementation of what we refer to as a *planfile*. Normally, Nimrod/G takes the Cartesian product of the declared parameters and creates a parametric

¹ 21 yearly computations * 12 months = 252 monthly computations.

hyperspace of Cartesian points², each representing a single task. To specify check-point dependencies in parametric spaces, we introduce the *seqameter* (Figure 4, line 3.3), a new type of *sequential* parameter. A *seqameter* behaves like a normal parameter, but its inclusion breaks a parametric instance into a set of temporally dependent sub-instances (see Figure 3). We have integrated *seqameters* in Nimrod/G’s planfile syntax, which provides explicit control to the user in setting the granularity of the checkpoints.

As can be observed in Figure 3, the splitting process introduced dependencies among the subtasks. Since the existing heuristics [18] in Nimrod/G were incapable of handling such dependencies, new scheduling heuristics were implemented to schedule tasks to resources based on:

- resource performance & network conditions;
- exploiting the parallelism among the application components, and
- respecting both task precedence and data affinity.

We implemented some existing algorithms, namely Sufferage [6], XSufferage [6] and a new LoadBalance heuristic, in the Nimrod/G scheduler. Each user-selected heuristic attempts to optimize task allocation with a specific objective function. Sufferage allocates a node to a task that would ‘suffer’ most if it is not allocated to that node. XSufferage is the cluster-level implementation of Sufferage. LoadBalance aims to balance the computational load across testbed resources. Sufferage and XSufferage require information about the network links and computational resource performance, which can be obtained from resource monitoring tools such as NWS [19] and Ganglia [20]. Apart from this information, statistical logs about the past performance of the resources can also be used in order to optimize task allocation. Nimrod/G uses both NWS traces and statistical logs to optimize task scheduling.

3.2 Data Management

Users are occasionally interested in performing operations on the state files, for example, backup instructions to store the file at a particular Storage Resource Broker (SRB) [21] location. Nimrod/G can be instructed to do this using the new planfile syntax. A specialized Nimrod task can be associated with these files, which can then be executed when a specific event occurs, such as file generation or copying (Figure 4, line 3.2). This function is declared

in the plan file (Figure 4, lines 5-6) as a normal task but it consists of some extra processing instructions.

If specialized operations are not associated with the state files, then those files are removed once the sub-task requiring them finishes its execution. Similarly, all other files, which are not required by any other task as an input, could be removed from the resources to free up some space. We therefore implemented garbage collection to remove all such files to comply with storage restrictions. This allowed unrestricted scheduling of multiple jobs on the same resource without the scheduler having to bother with space constraints. Collecting garbage in this manner solved data management problems to a large extent.

3.3 Trust Management

Coupling resources belonging to multiple VOs together provides a larger resource pool than what would be available in a single VO. However, in multi-VO Grid testbeds, the scheduler might allocate a task to a resource within a VO that is different from that of its parent. Since the parent’s state file must be passed on to its child subtask, a third party transfer across resources in different VOs will become necessary. As described earlier, third party data transfers across VOs cannot proceed unless trust conflicts are first resolved. In principle, if the user is authorized to access resources from two different VOs, then it should be possible to perform third party data transfers between the resources. In the Globus Security Infrastructure (GSI) [1], a client must possess legitimate credentials that are validated against the CA trusted by the resource to be accessed. To access resources from different VOs, and thus, across different CAs, distinct credentials valid for each CA are required. Each valid credential is the basis for a *proxy certificate* that authenticates the user and authorizes access to resources of the associated VO.

Proxy certificates, which can be used to perform third party transfers, are available to Nimrod/G. However, Nimrod/G supports a range of different Grid middleware apart from Globus. For example, Nimrod/G currently supports file transfers across resources managed by Condor[13], PBS[22], Sun GridEngine[23], Bproc[24] and Fork. We therefore implement the third-party transfer mechanism within Nimrod/G to transcend the heterogeneity among transfer methods that might otherwise be involved. Third party transfers are implemented using *Nimrod agents* and *Nimrod Remote File Servers*, and secured through public key authentication. Nimrod/G manages the execution of tasks using Nimrod agents that run on the resource where a user’s tasks are. Nimrod agents also manage data transfers normally required by tasks,

² A Cartesian point can be defined as (a, b) where $a \in A$ and $b \in B$, A and B being the two parametric sets.

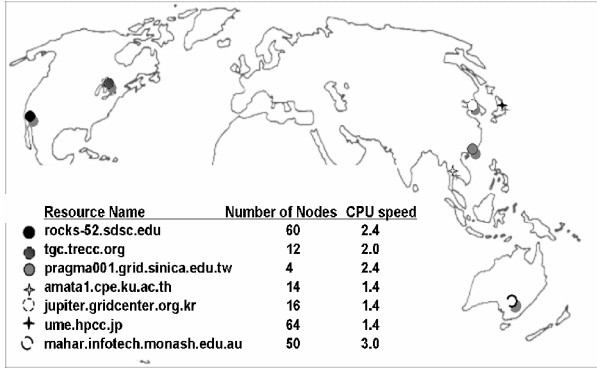


Figure 5. Testbed configuration.

however, third party transfers require Nimrod remote file servers. To validate transfer requests from Nimrod agents involving a given resource, a remote file server is launched on it. Distinct session keys for each resource, say x and y , are generated and maintained at the user's local site, in the Nimrod database. In order to perform the transfer from resource x to y , the session key required to access resource y is copied to x . The Nimrod agent on x then uses the copied key, on behalf of the user, to validate the request with the Nimrod remote file server on y . Once validated, the Nimrod remote file server facilitates the transfer operation.

In this manner, we provide a secure third party mechanism across resources from different VO's, which may support different standards. The actual

transfer method relies on Nimrod/G, without having to introduce another delegation layer such as WebCom's CCA [25]. Please note that the user does not need to install Nimrod/G on remote resources. However, Nimrod/G performs these operations by launching remote agents and file servers on distributed resources. We are thus able to use resources belonging to multiple VO's as a single "Grid," leveraging its distributed storage capacity.

4 Experiments and Results

Here we present the results of the Savannah experiment on the PRAGMA [21] testbed (Figure 5), consisting of 7 resources from 6 different countries. Each resource had its own CA domain and showed dynamic availability patterns during the experiment.

4.1 Task Management

Figure 6 shows the state transition diagrams of two sub-computations. The upper half of each state node shows the allocated resource, whereas the lower half describes the current state of the job. The sub-computations were migrated from one resource to another mainly due to:

- Grid resource failures, when the resource that was allocated to the job eventually failed;
- Input file copy failures, when the file copy failed due to resource failure or network failure;

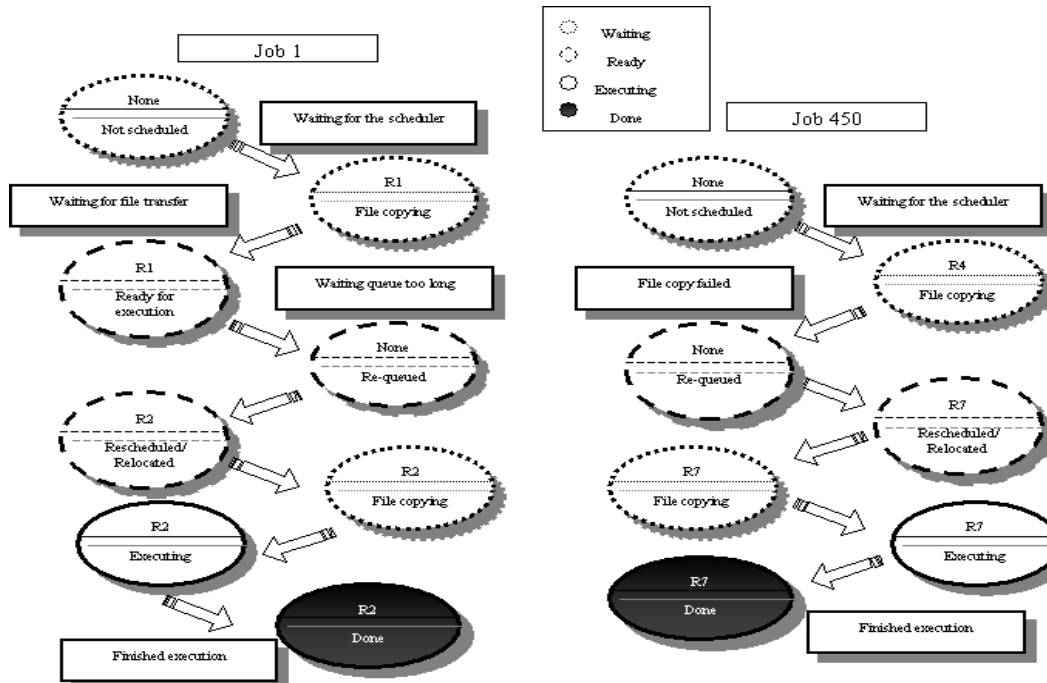


Figure 6. Diagrams of two sub-computations

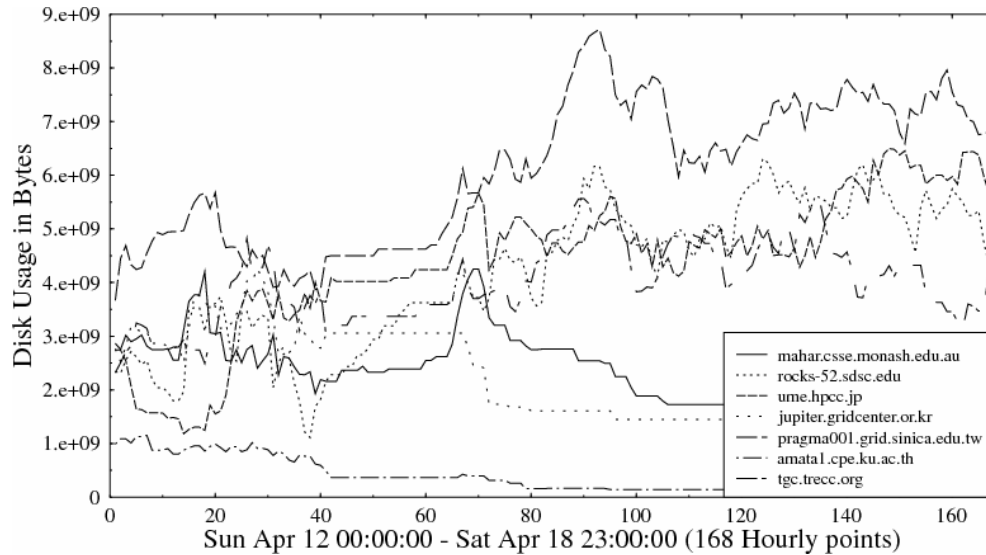


Figure 7. Disk space usage patterns.

- Resource unavailability, when the front node's job queues were too long or the resource was too busy;
- Job failures, where the job failed due to missing libraries or some unexpected runtime error and
- Disk space shortage, if the input files could not be copied to the scheduled resource due to disk space limitations.

However, in every case, Nimrod/G migrated the jobs to other available resources to recover from these faults. Furthermore, the heuristics implemented in Nimrod/G provides a range of objective functions to optimize scheduling.

Such fault recovery and adaptability are difficult to achieve if the applications, of the scale described in this paper, were executed under the conventional parameter sweep model.

4.2 Data Management

Figure 7 shows the disk space usage of the experiment running on different resources. The graph, which covers a few days of the whole experiment, illustrates dynamic data management activities as disk usage grows and shrinks. An increase in the y-coordinate represents data accumulation, whereas a decrease represents file removal. In the savannah experiment, the disk space usage was directly proportional to the number of jobs executing on the resource. Therefore, a low y-coordinate (disk usage) value in the graph means that fewer jobs executed on that resource at that moment. An oscillation shows that the data is being generated and removed uniformly. The job vs. data relationship is important in order to understand the following behavioral analysis.

It can be observed that before the 70th hour (approximately), all resources had a smooth

oscillatory behavior, but the disk usage of `jupiter.gridcenter.or.kr`, `mahar.csse.monash.edu.au` and `amata1.cpe.ku.ac.th` decreased afterwards, because these resources became unavailable. The constant disk usage at the end of the graph is due to the presence of common files, which were retained on the resources for subsequent use by other computations. Due to resource unavailability, the jobs were distributed among other available resources, which is evident from the increase in their disk usage patterns. Note how resources such as `ume.hpcc.jp` have shown a more noticeable increase in disk usage compared to others. This happened because these resources had a higher number of computational nodes available, which made them a preferable choice for job allocation. Note as well that the oscillation in disk usage curves became quite uniform after the 100th hour when the load was distributed again properly. This also illustrates the effectiveness of the garbage collection technique.

4.3 Trust Management

As mentioned before, we had 90 parametric instances of the main model, which were broken into a sequence of 252 sub-computations and interconnected by the state files (see Figure 3). In this setup, every successive sub-computation required the state file generated by its immediate predecessor as its input for execution. Each line in Figure 8 represents the first 20 sub-computations of randomly chosen parametric instances. It can be observed that sub-computations, belonging to a particular instance, were scheduled across 7 different resources with exclusive CA domains. The situation required the state files to be copied accordingly (more resources were added at the

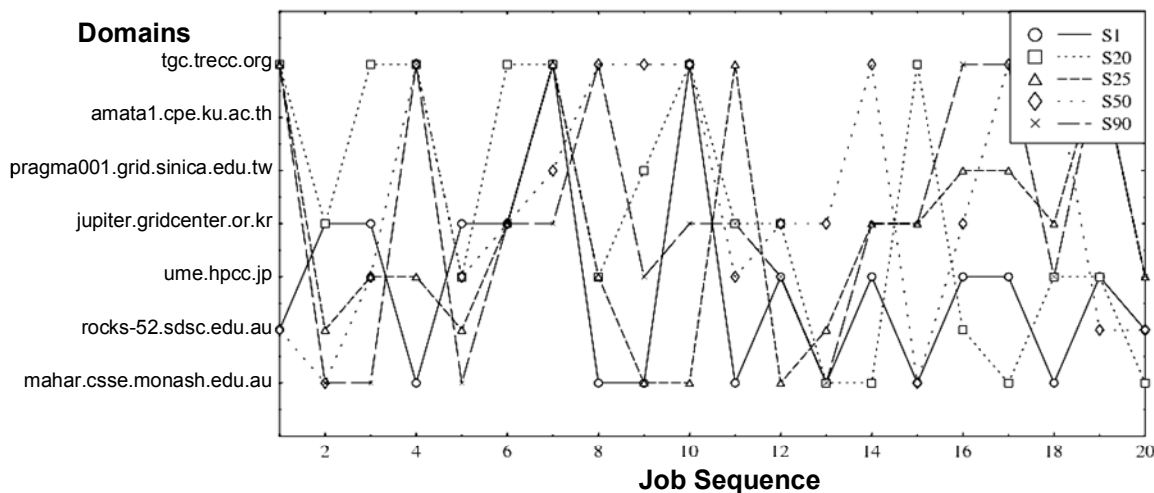


Figure 8. Inter-domain third party file transfers.

later stage of the experiment). For example, to execute the 12th sequence of the 25th parametric instance on mahar.csse.monash.edu.au, the state file generated by the 11th sequence was copied from domain A (tgc.trecc.org) to domain B (mahar.csse.monash.edu.au). We performed file transfers using our third party communications mechanism, which allowed us to maximize the usage of the multi-VO Grid testbed.

5 Related Work

A number of projects complement our work. The Apples Parameter Sweep Template (APST) is a parametric modeling tool, which allows the scheduling of complex workflows on the Grid, specified using large APST XML descriptions. Such descriptions are unlike the simple planfile syntax developed for Nimrod/G and subsequently used by GridBus. The APST scheduling model also assumes unlimited disk storage and single VO testbeds. GridBus' core technology is quite similar to that of Nimrod/G, but the former assumes unlimited disk storage and single-domain testbeds. ICENI provides a dynamic service-oriented framework for Grid environments that are restricted to single domains, without any task-splitting and garbage collection.

6 Conclusion and Future Work

In this paper, we have presented new task, data and trust management techniques to deploy large-scale applications on the multi-VO Grid testbed. These techniques were implemented during the first phase of the experiment involving the execution of large PSAs to study how the burning of savannah affects the

climate of Northern Australia. The implemented techniques include:

- A task-splitting and checkpointing approach to break up large PSAs into a sequence of dependent sub-tasks for better fault recovery and improved application performance.
- A garbage collection technique to delete unnecessary data to comply with storage limits.
- A third-party file communication technique which delegates trust to agents deployed across multiple VOs to perform cross-domain file transfers.

Results show that Nimrod/G migrates sequential sub-computations to better resources when failures occur, while still executing as many tasks in parallel as possible. Moreover, disk usage patterns on computational resources illustrate that Nimrod/G efficiently co-schedules multiple computations on a single resource. However, Nimrod/G restricts the amount of cumulative data generated by these computations within permitted storage limits. Further results show that whilst migrating the downstream sub-computations across resources of multiple VOs, Nimrod/G effectively copies the state files along with other input files required by individual sub-computations to appropriate resources.

Our future work will include:

- specialized CPU and network state monitoring to optimize task scheduling;
- an intelligent backup and restore mechanism to minimize state-backup costs and to improve fault recovery; and
- implementation of novel clustering-based heuristics to optimize scheduling of dependent tasks.

7 Acknowledgement

This work is supported by the Australian Research Council under an e-Research special initiative grant. We wish to thank Amanda Lynch and her colleagues for their assistance in running the climate study. We also acknowledge the CSIRO Division of Atmospheric Research who provided the computational models. We thank our PRAGMA partners for the resources used in the testbed.

8 References

- Abramson, D., et al., *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid ?*, in *International Parallel and Distributed Processing Symposium*. 2000.
- Amsden, A., et al., *Kiva: A computer program for 2- and 3-dimensional fluid flows with chemical reactions and fuel sprays*. 1985, Los Alamos National Laboratory.
- Basney, J., M. Livny, and P. Mazzanti, *Harnessing the Capacity of Computational Grids for High Energy Physics*, in *In Conference on Computing in High Energy and Nuclear Physics*. 2000.
- Rogers, S., *A Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations with Artificial Compressibility*. AIAA Journal, 1995. **33**(10).
- Stiles, J.R., et al., *Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes*. Computational Neuroscience, 1998: p. 279-284.
- Casanova, H., et al. *Heuristics for Scheduling Parameter Sweep Applications in Grid Environments*. in *In Proceedings of the 9th Heterogeneous Computing Workshop (HCW00)*. 2000.
- Buyya, R. and S. Venugopal, *The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report*. In *Proceedings of the First IEEE International Workshop on Grid Economics and Business Models*. 2004: New Jersey, USA.
- Y. Tamir, C.H.S., "Error recovery in multicomputers using global checkpoints". 1984.
- Luiz Meyer, J.A., Mike Wilde, Marta Mattoso, Jan Foster "Planning spatial workflows to optimize grid performance". in *Proceedings of the 2006 ACM symposium on Applied computing, Distributed systems and grid computing (DSGC)*. 2006. Dijon, France.
- Cindy Zheng, D.A., Peter W. Arzberger, Shahaan Ayyub, Colin Enticott, Slavisa Garic, Mason J. Katz, Jae-Hyuck Kwak, Bu-Sung Lee, Philip M. Papadopoulos, Sugree Phatanapherom, Somsak Sriprayoosakul, Yoshio Tanaka, Yusuke Tanimura, Osamu Tatebe. *The PRAGMA Testbed - Building a Multi-Application International Grid*. in *Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06)* 2006. Singapore.
- McGregor, J.L., Nguyen, K. C., and Katzfey, J. J. *Regional climate simulations using a stretched-grid global model*. in *Research activities in atmospheric and oceanic modelling*. . 2002.
- Geneva: H. Ritchie (ed.). (CAS/JSC Working Group on Numerical Experimentation Report.
- Beringer, J., et al. *The Savanna Fire Experiment (SAFE)*. [cited; Available from: <http://www.arts.monash.edu.au/ges/research/climate/fire/index.html>].
- Buyya, R., D. Abramson, and J. Giddy, *An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications*, in *Workshop on Active Middleware Services (AMS 2000)*, (in conjunction with Ninth IEEE International Symposium on High Performance Distributed Computing). 2000, Kluwer Academic Press.
- Abramson, D., R. Buyya, and J. Giddy, *A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker*. Future Generation Computer Systems, 2002. **18**(8).
- M. Y. Gulamali, A.S.M., S. J. Newhouse, and J. Darlington. *Using ICENI to run parameter sweep applications across multiple Grid resources*. in *Case Studies on Grid Applications Workshop, Global Grid Forum 10*. March 2004. Berlin, Germany.
- Casanova, H., et al., *The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid*, in *In Proceedings of the Super Computing Conference (SC'2000)*. 2001.
- Casanova H. , J.H. *APST XML man page*. 2001 [cited; Available from: http://grail.sdsc.edu/projects/apst/apst_xml_man.html].
- Buyya, R., D. Abramson, and M. Murshed, *A Deadline and Budget Constrained Cost-Time Optimization Algorithm for Scheduling Task Farming Applications on Global Grids*, in *The 2002 International Conference on Parallel and Distributed Processing Techniques and Applications*. 2002: Las Vegas, Nevada, USA.
- Rich Wolski, N.T.S., Jim Hayes. *The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing in Future Generation Computer Systems*. 1998.
- Implementation, a.E.M.L.M.U.o.C. *The Ganglia Distributed Monitoring System*.
- Chaitanya Baru, R.M., Arcot Rajasekar, Michael Wan. *The SDSC Storage Resource Broker*. 1998.
- Portable Batch Scheduler*.
- Gentzsh., W. *Sun grid engine: Towards creating a compute power grid*. in *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*. 2002.
- Hendriks, E.A. *The Beowulf distributed process space*. 2000.
- David W. O'Callaghan, B.A.C. "Bridging Secure WebCom and European DataGrid Security for Multiple VOs over Multiple Grids". in *Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04) - Volume 00*. 2004: IEEE Computer Society