Using Markov Chain Analysis to Study Dynamic Behaviour in Large-Scale Grid Systems

Christopher Dabrowski and Fern Hunt
National Institute of Standards and Technology
Gaithersburg, MD, USA 20899
[Christopher.Dabrowski, Fern.Hunt]@nist.gov

Abstract

In large-scale grid systems with decentralized control, the interactions of many service providers and consumers will likely lead to emergent global system behaviours that result in unpredictable, often detrimental, outcomes. This possibility argues for developing analytical tools to allow understanding, and prediction, of complex system behaviour in order to ensure availability and reliability of grid computing services. This paper presents an approach for using piece-wise homogeneous Discrete Time Markov chains to provide rapid, potentially scalable, simulation of large-scale grid systems. This approach, previously used in other domains, is used here to model dynamics of large-scale grid systems. In this approach, a Markov chain model of a grid system is first represented in a reduced, compact form. This model can then be perturbed to produce alternative system execution paths and identify scenarios in which system performance is likely to degrade or anomalous behaviours occur. The expeditious generation of these scenarios allows prediction of how a larger system will react to failures or high stress conditions. Though computational effort increases in proportion to the number of paths modelled, this cost is shown to be far less than the cost of using detailed simulation or testbeds. Moreover, cost is unaffected by size of system being modelled, expressed in terms of workload and number of computational resources, and is adaptable to systems that are non-homogenous with respect to time. The paper provides detailed examples of the application of this approach.

Keywords: Grid computing; Perturbation analysis; Discrete Markov chain; Piece-wise homogenous Markov chain.

1 Introduction

The long-term continued commercial success of grid technology will likely depend on emergence of large-scale, decentralized grid systems in which large numbers of service providers and consumer clients enter into *service-level agreement*s (SLAs) (Andrieux et al. 2007) to allocate grid resources. Here, as in other large-scale systems with decentralized control, the interactions of many consumers and providers can lead to emergent global system behaviours that result in unpredictable, often detrimental, outcomes (Mills and Dabrowski 2006). The movement toward realization of large-scale grid systems is evident in recent developments such as commercial cloud computing, in which mass computing services are made available for sale on-line. Clouds and other commercial developments likely foreshadow eventual creation of grid compute economies that operate on market principles. Having in place analytical tools to allow understanding, and prediction, of complex system behaviour will be necessary to ensure availability and reliability of grid services in economic settings.

For these reasons, the development of analytical tools that take into account complex systems behaviour will be a necessity. In particular, tools that can predict the impact on overall system performance of changes to key system parameters will be of particular importance. Previous researchers have used simulation to study behaviour of grid systems that utilize different economic strategies (Chun and Culler 2002, Yeo and Buyya 2005, Mills and Dabrowski 2008). Studies of failure scenarios in grid system such as (Mills and Dabrowski 2006) have shown that small variations in key variables can lead to alternative execution paths that yield large differences in overall system performance. Although more practical than using operational grid systems as testbeds, large-scale simulations that attempt to accurately reproduce system structure and component behaviour are often a computationally expensive proposition when many alternative execution paths must be considered. Moreover, computational expense increases dramatically with increase in model size, a critical factor for analysing grid systems which, even now, can consist of more than 10^5 computing resources (Carr 2006, Raffo 2006).

To remedy this situation, this paper presents an approach in which discrete time Markov chain analysis is combined with a form of rapid, scalable, simulation. This approach, previously used in other areas, is used here to model dynamics of large-scale grid systems. In this approach, a state model of the system is first derived by observing system operation and then converted into a succinct Markov chain representation in which model scale is reduced by taking advantage of the stochastic characteristics of this model. The resulting model is expressed as a set of transition probability matrices (TPMs) that succinctly summarize system dynamics over different time periods. The TPMs represent an execution path that can be changed by altering, or *perturbing*, the values of individual transition probabilities in the TPM. By systematically perturbing combinations of transition probabilities, it is possible to model alternative execution paths, each of which lead to a different evolution of a grid system over time. Among these are execution paths

where failure to meet fundamental guarantees of service causes system performance to significantly degrade.

The approach presented in this paper allows expeditious investigation of a large number of alternative system execution paths and identification of paths, or scenarios, in which failure to meet service guarantees adversely affects overall system performance. In this way, Markov chain analysis can be used to predict how a larger system will react when key service guarantees are not met. Though computational effort increases in proportion to the number of paths modelled, we find that the cost of using Markov chains is far less than the cost of searching the same problem space using detailed, large-scale simulation or testbeds. Moreover, computational cost is unaffected by size of system being modelled, where size is expressed in terms of workload and number of computing resources. The approach can also be adapted for cases in which transition probabilities change with (e.g. are non-homogenous with respect to) time.

The plan of this paper is as follows. Section 2 summarizes previous work on Markov chain analysis and related techniques in distributed systems. Section 3 more precisely describes the problem being investigated using Markov chain analysis. The section defines fundamental guarantees of service that large-scale grid systems will need to provide to their customers and which are the basis for the analysis in this paper. Section 4 describes the state model for our grid system, how this model is extended to be a Markov chain model, and how the Markov chain is used for simulation. This section describes how the size of the model representation is reduced and adapted for situations where the Markov chain is non-homogenous with respect to time. Section 5 describes the method of perturbing the Markov chain TPM to simulate alternative execution paths that violate service guarantees defined in section 3. Section 6 presents results of using the methods described in sections 4 and 5 to predict system evolution and compares these results to those produced by more detailed simulation. Section 7 presents conclusions.

2 Previous Work

This section briefly reviews work on use of Markov chains, focusing on two outstanding problems: methods to reduce model size and perturbation analysis techniques that reduce the size of the perturbation space. Discrete Time Markov chain (DTMC) analysis is a well established analytical tool that has been applied to study dynamic system behaviour in a variety of real-world domains. Markov chain analysis has long been used in manufacturing for problems such as transient analysis of dependability of manufacturing systems (Zakarian and Kusiak 1997), split and merge production line processes and part quality defects (Li et al. 2008). Markov chain analysis has been used to model mean time to failure in communications networks (Cassandras, Lee, and Ho 1990), link reliability (Balakrishnan and Reibman 1994), as well as to examine fault-tolerance and performance in multi-processor computer architectures (Aupperle and Meyer 1991, Chiola et al. 1993), real-time process control systems (Trivedi, Ramani, and Fricks 2003), and software systems (Laprie and Kanoun *1992*, Goseva and Trivedi 2001). In grid computing, Markov chains have been used to model workload for scheduling (Song, Ernemann, and Yahyapour 2004) and load balancing (Akioka and Muraoka 2003). However, unlike these efforts or those that quantitatively estimate performance or reliability, this work uses Markov chain modelling to understand alternative system behaviours that may occur as a consequence of significant system-wide events or decisions: in this case, the failure to meet fundamental service guarantees for grid systems.

The combinatorial increase of the number of states in DTMC models for large problems has long been widely recognized as a barrier to practical use of Markov chain analysis. To solve this problem, the concept of lumping states with similar characteristics into larger aggregated units was introduced (Kemeny and Snell 1976) and has been worked on extensively since. Various lumping approaches have been explored that use model structure and symmetry to reduce size (Siegle 1992, Buchholz 1995, Nicol, Sanders, and Trivedi 2004). Other methods for reducing model size are based on group-theoretic concepts (Aupperle and Meyer 1991), Stochastic Activity Nets (Sanders and Meyer 1991), stochastic coloured nets (Chiola et al. 1993), use of reward variable structures to identify symmetries (Obal and Sanders 2001), and use of eigenvector equivalence classes to partition a Markov state space into lumps (Jacobi and Gornerup 2007). Fortunately, in the model we present, the number of states is readily reducible using the stochastic characteristics of Markov chains, described in section 4. While the number of states in our model did not prove to be a barrier, the size of the perturbation analysis problem did.

Perturbation analysis of discrete time Markov chains has been the topic of theoretical work in the last three decades (Schweitzer 1968). Like the problem of model size, the size of a typical perturbation space may quickly become computationally intractable, if there are many combinations of alternative system variable values to consider. To attack this problem, some researchers (Ho 1985, Suri 1989) have advanced the idea of perturbation analysis of discrete event systems by calculating system performance gradients that are based on key decision parameters. This approach estimated the sensitivity of changes to decision parameters in order to optimize system performance. Gradient-based approaches were seen to have the potential to reduce the size of the perturbation space because they needed to observe as few as one execution path of a system.

This approach was adapted for Markov chains by estimating gradients for alternative execution paths (Ho and Li 1988, Suri 1989). More recently the gradient-based approach was extended to reduce problem size in Markov models by grouping state transitions on the basis of events in order to evaluate control policies (Cao 2005, Cao and Zhang 2008). This approach was believed to scale with the number of events and size of the system. However, in this and earlier work, determination of performance vectors and efficient gradient calculations were issues that were not fully resolved. Further, not all problems were reducible to a form which allowed tractable calculation of gradients for specific control policies. While gradient-based perturbation algorithms have demonstrated potential as efficient tools for analysis of some complex systems, they also introduce not inconsiderable computational issues and were found not

to be applicable to all Markov problems. Moreover, the gradient-based approaches appear more geared to optimization problems that depend on relatively few system parameters, rather than the more general problem of assessing alternative execution paths.

Instead, the approach presented in this paper avoids the computational difficulties of gradient-based methods. The potential problem of size in Markov models is mitigated through a straightforward, concise problem representation that takes advantage of the stochastic characteristics of Markov chains and an intuitive, limited search strategy. While this approach does not completely solve the issue of problem size, we show Markov chain analysis yields comparable results at a fraction of the computing cost of a large-scale simulation and provides a viable analytical tool for study of system dynamics.

3 Questions to be Answered Through Perturbation of Markov Chains

It is convenient to organize this analysis on the basis of basic guarantees of service that grid systems must provide to their users. These guarantees constitute basic grid system requirements, which if not met, may render a grid system useless. The extent to which a grid system fulfils, or does not fulfil, these guarantees impacts overall system performance. The ability of Markov chain analysis to efficiently predict system behaviour if the guarantees are not met is good way to gauge the usefulness of Markov methods. Three guarantees may be described.

First, a grid system must guarantee that current information about what grid computing services exist is available to users. In grid systems, this guarantee is fulfilled through service discovery mechanisms that locate needed services and make information about them available to users. The *service discovery guarantee* refers to the ability of a grid system to provide necessary information about grid computing services, including relevant updates, which users require to make decisions.

Second, if a user has found a needed service, the service is available (not reserved for other tasks), and the user is qualified to use the service, then the grid system should allow the user to engage that service. This is called the *service engagement guarantee*. To be qualified, the user must have security and administrative access, and be able to afford the service. The service engagement guarantee is meant to ensure that users and providers of services who logically should cooperate, in fact do so. In most cases, engagement of a service is signified by the formation of an SLA, which reserves the service for the user for a fee. The third guarantee is the *service fulfilment guarantee*, which simply states that once a service has been engaged, i.e. the SLA is in place, the terms of the agreement should be fulfilled by both provider and user.

Understanding and predicting the consequences of not fulfilling these guarantees is an important analysis problem. Particularly important is understanding how the performance of a grid compute economy is affected as the extent of guarantee fulfilment decreases. Administrators as well as providers and users need to understand how different levels of non-fulfilment of each guarantee affect a grid system. At what point of incremental increase of guarantee non-fulfilment does system performance begin to degrade rapidly? What specific actions by providers or consumers affect non-fulfilment of a particular guarantee? Answering questions like these by taking an actual production system offline to use as a testbed is impractical for obvious reasons. As indicated above, simulation has been used successfully to estimate impacts of failure scenarios. However, if simulation requires many repetitions using a detailed compute-intensive model to examine the effect of many system parameters, the analysis may either take considerable time, be limited to a restricted number of alternative execution paths, or both. Markov chain analysis thus provides a viable alternative for obtaining understanding of effects of not fulfilling grid service guarantees. The rest of the paper describes the basic approach and shows initial results.

4 The Markov Chain Model

The behaviour of a large-scale grid system can be modelled in terms of the computing tasks executing in the system at any time. Each task progresses through a life cycle in which it is first submitted by a user, service providers are discovered to run the task, an SLA is negotiated with selected provider(s), and the task either executes to completion or fails. The state of the grid system can be described by the states of all the tasks that are in the system at a particular time. This section first describes the state transition model for a single task and then shows how the aggregate of many task states can be represented in a concise Markov chain model. This model is then elaborated to represent a piece-wise homogenous Markov chain that allows the dynamics of the grid system to be studied over different time periods.

4.1 Representing a Task Lifecycle as a State Model

The lifecycle of an individual task can be represented in seven states, shown in Figure 1. This model is derived from a large-scale simulation (Mills and Dabrowski 2008) that studies operation of a grid over an 8-hour day. Three states in the model presented here—*Discovering*, *Negotiating*, and *Monitoring*—can be further decomposed into sub-states. The decompositions yield 27 additional states, resulting in a larger model with a total of 34 states (described in Dabrowski and Hunt 2008). Because the 34-state model can be aggregated into the simpler seven-state model, the latter is used in the rest of the paper.

The high-level model representation may be described as follows. In the *Initial* state, a task has not yet entered the grid system. Each task is assigned an arrival time and deadline from exponential distributions (Mills and Dabrowski 2008). At the arrival time, the task automatically transitions to the *Discovering* state. In *Discovering*, the task client attempts to discover eligible providers with sufficient computing resources to execute the task. After discovery actions conclude, the task may either transition to *Negotiating* or *Waiting*. Tasks enter the *Negotiating* state at regular intervals. A task that has completed *Discovering* and found at least one provider enters *Negotiating* if the interval has elapsed; otherwise it goes into the *Waiting* state. In *Negotiating*, clients rank discovered providers that they are qualified to use, e.g., can afford, on the basis of anticipated cost. They contact

each provider, one at a time, and offer an SLA to execute the task for a fee. Once an available provider accepts the offer, negotiation ceases and the task enters the *Monitoring* state, during which the task is either blocked on an execution queue or executing. If negotiations fail (i.e., no provider can be found to accept an SLA), the task goes from the *Negotiating* state to either *Discovering* or *Waiting*. As in *Negotiating*, a task enters *Discovering* at regular intervals. If negotiations fail, a task transitions from *Negotiating* to *Discovering* if the start time has arrived. Otherwise it transitions to *Waiting* and remains until the next *Discovering*, or *Negotiating*, start time.

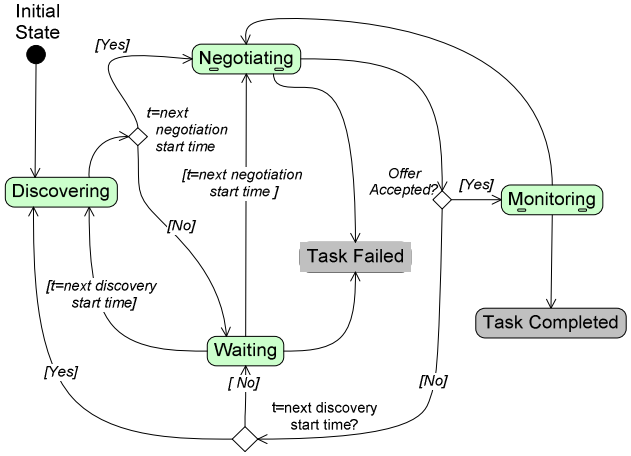

Figure 1. State model of the grid compute economy.

A task that is in *Monitoring* state enters the *Completed* state if task execution is successful. If execution fails, the task re-enters the *Negotiating* state. Tasks may also transition to the *Failed* state from either the *Negotiating* or *Waiting* states. This occurs when it becomes impossible to complete the task by its deadline. Both *Completed* and *Failed* are terminal states from which tasks cannot leave once they enter.

4.2 Evolving the State Model to a Markov Chain Model

A Markov chain has the property that the probability of transition between any two states depends entirely on the circumstances in the state from which the transition originates and not on the previous history of the process. More formally, given a sequence of states X_1, X_2, …… X_n, the Markov Property is given as:

$$\Pr\begin{pmatrix} X_{n+1}=x \mid \\ X_n=x_n,\ldots,X_1=x_1 \end{pmatrix} = \Pr\begin{pmatrix} X_{n+1}=x \mid \\ X_n=x_n \end{pmatrix} \qquad (1)$$

The state model depicted in Figure 1 satisfies the Markov property. Careful review of the preceding description shows that the decision to transition to another state depends only on the circumstances of the state the task is currently in. These circumstances include whether a time interval has elapsed, an SLA has been secured, task execution has succeeded or failed, etc.

In a Markov chain, probabilities are associated with transitions between states. To calculate state-to-state transition probabilities, transition frequencies are first summed over a simulated an eight-hour day using the large-scale model (Mills and Dabrowski 2008). This is done by determining where state transitions occur in the model code and inserting counters at those places. In our experiments, frequencies were summed for all state transitions over 50 repetitions of a 36000 s period (10 hours: 8 hours + two extra hours for late tasks) with a 75% workload level. State transition probabilities were derived as follows. Given states s_i, s_j, $i, j = 1\ldots n$ where n=7, p_{ij}, is the probability of transitioning for state i to state j, written as $s_i \rightarrow s_j$. This probability is estimated by calculating the frequency of $s_i \rightarrow s_j$, or f_{ij}, divided by the sum of the frequencies of s_i to all other states s_k, or

$$p_{ij} = \frac{f_{ij}}{\sum_{k=1}^{n} f_{ik}} \qquad (2)$$

Here i and j may be equal, to allow for transition of a state to itself, or *self-transition*. A *self-transition* occurs when a task remains in a state longer than a specified interval (equal to a Markov simulation discrete time step, h, described below). The resulting TPM is a 7 × 7 stochastic matrix, shown in Figure 2. Here rows stand for the state the transition originates from, or *from state*, and columns represent states the transition goes to, or *to state*. Each cell in a TPM represents a p_{ij}, where i and j are from and to states, respectively. As in any stochastic TPM, the transition values of all columns in a row must sum to 1.0. The only exception to this procedure involved arrival of tasks in the grid system, described above. Here, the Markov chain process was altered to reproduce exactly the exponential arrival times of the large-scale simulation.

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0.9697	0	0.0303	0	0	0	0
Waiting	0	0.8363	0.0673	0.0918	0	0	0.0046
Disc	0	0.0355	0.6714	0.2931	0	0	0
Ngt	0	0.4974	0.0182	0.2882	0.1961	0	0.0001
Mon	0	0	0	0.0003	0.9917	0.0080	0
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	1.0

Figure 2. Summary stochastic TPM. This TPM is a weighted average of five TPMs for equal time period divisions over the 36000 s duration. Individual p_{ij} from the five periods are weighted by the relative number of transitions in their respective periods. See Appendix for time period matrix set.

The Markov chain and related TPM can be further classified. Careful analysis of the description of the state model in section 4.1 and the structure of the matrix in Figure 2 shows that tasks can enter the *Discovering*, *Waiting*, *Negotiating*, and *Monitoring* states multiple times, but always remain temporarily. At some point they enter either the *Completed* or *Failed* state, where they remain permanently, or are *absorbed* (These states are *absorbing* states, in which only self-transitions are possible). A Markov chain with these characteristics is called an *absorbing chain* (Kemeny and Snell 1976) that can be divided into a *transient* part (the *Discovering*, *Waiting*, *Negotiating*, and *Monitoring* states), and an absorbing part (with two absorbing states, *Completed* and

Failed). This characterization is born out in the Markov chain simulation described below. While the summary TPM in Figure 2 is useful for illustrating concepts and for certain analytical studies of the system, it requires further elaboration to model non-homogeneity.

4.3 Reducing Model Size and Handling Time Non-Homogeneity

In the large-scale simulation at a 75% workload, there are typically over 400 tasks, each of which progresses through the seven states. With such numbers, it is impossible to model a system state in which all tasks are tracked simultaneously. However, the stochastic nature of Markov chains allows one to consider the distribution of the 400+ tasks among the seven states. It is easy to see that a system-wide state can be represented in terms of the proportion of tasks that are in each state. In this way, it is possible to represent the system state as a seven-element vector in which the value of each vector element represents the proportion of tasks in the related state. This method of system state representation is a simplification that together with the combination of 34 states into seven, described above, facilitates problem analysis.

In the large-scale simulation, task submission times varied exponentially over the simulated day, with mean task start time at t=3600 s (end of the first hour). This distribution resulted in different workload levels at different times in the day and caused transition probabilities over the 36000s-period to vary. Therefore, different TPMs were actually in force at different times, making the system non-homogenous with respect to time. For this reason, more accurate simulation results for the transient behaviour of the system were obtained by creating time-period partitions and computing a separate TPM for each period. In this experiment, frequencies were summed separately for five time periods of 7200 s each[2]. These matrices, shown in the Appendix, allow a representation of our model as a *piece-wise homogenous* Markov chain having a bounded number of pieces (Rosenberg, Solan, and Veille 2004) corresponding to the time periods. To produce the summary TPM, shown in Figure 2, the individual transition probabilities in the TPMs for these five periods were weight averaged on the basis of the relative transition frequencies in each period. In the summary matrix, each weight-averaged probability of transition, p_{ij}, is computed as follows

$$p_{ij} = w_i^1 p_{ij}^1 + w_i^2 p_{ij}^2 + \ldots . w_i^{nper} p_{ij}^{nper} \quad (3)$$

in which each w_i^l represents the weight for row i in time period l, $l \in \{1..\ nper\}$ where $nper$=5. Each w_i^l is computed by

$$w_i^l = \sum_{1 \le j \le n} f_{ij}^l \Big/ \sum_{1 \le tp \le nper} \sum_{1 \le j \le n} f_{ij}^{tp} \quad (4)$$

where each f^{tp}_{ij} is the frequency of transition from state i to state j in time period tp and n is the dimension of the matrix (n = 7). The five time-period matrices more accurately captured system dynamics over time than did the summary TPM in Figure 2 and were therefore used in the Markov simulations described below.

4.4 Using a Sequence of Markov Chain TPMs to Simulate a Dynamic System

A well-known use of stochastic TPMs in a Discrete Time Markov chain is to describe how a dynamic system changes over time in discrete time steps, where each step represents a fixed time duration. In this experiment, a discrete time step is chosen to represent 85 s, or h = 85 (which also is the time duration for a self-transition, discussed in section 4.2)[3]. Hence, if a time period covers a duration of d_{period} = 7200 s, each of the five time-period matrices represent S= d_{period} /h steps or 85 steps.

As indicated above, the system state can be summarized in a vector v, where each element represents the proportion of tasks in one of the seven states. Using equation (5), a vector v_m, which represents the system state at time step m, is multiplied by the TPM Q^{tp} for the applicable time period tp to produce a new system state v_{m+1}, to evolve the system over a single discrete time step.

$$(Q^{tp})^T * v_m = v_{m+1}, \text{ where } tp = integral\ value\ (m/S) + 1 \quad (5)$$

where T indicates a matrix transpose. Starting with v_0, which represents a system state with a value of 1.0 for the *Initial* state and 0 for all others (e.g, all tasks are in *Initial*), equation (5) is repeated for 339 time steps (representing 28,800 s or the simulated 8-hour day). This results in a system state vector, v_{339}, in which the sum of the proportion of tasks in the *Completed* and *Failed* states approaches 1, while other states fall to 0. A goal of Markov chain analysis is to execute this procedure with a set of time period TPMs derived from a real-world system (or, in this case, the large-scale simulation) in order to approximate the operation of that system. Figure 3 compares the proportion of tasks in the *Completed* and *Failed* states in the Markov chain simulation over 339 steps with the proportion of tasks in these states in the large-scale simulation as it executes for 28,800 s.

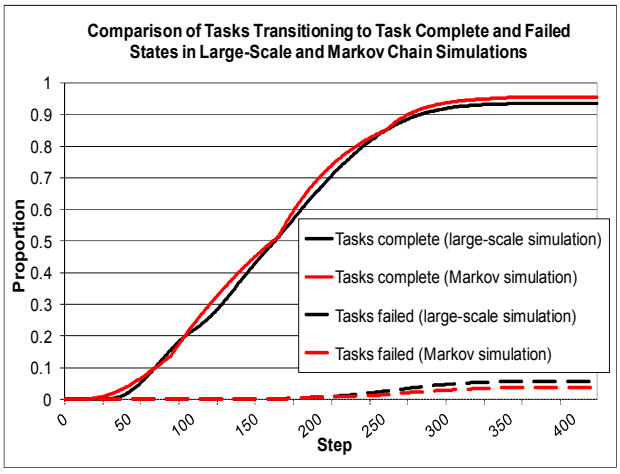

Figure 3. Change in *Completed* and *Failed* states over time in large-scale and Markov chain simulation.

[2] Different numbers of time periods were attempted, including three, 10, and 15; however, five provided the most accurate results. Devising a method of selecting an optimal number of time periods is left for future work.

[3] Different values for h were also tried. A method of selecting h is left for future work.

This paper argues that a piecewise homogenous Markov chain can approximate the transient behaviour of a real-world grid system (for which the large-scale simulation is a proxy). By applying (5) to a set of perturbed TPMs to simulate alternative evolutions of a grid system, one can model, or predict, the effects of changed system capabilities, undesirable behaviours, and events of interest. In this study, we are interested in perturbing selected rows of the TPM set to represent changes in the ability of the system to fulfil the three grid system guarantees described in section 3, and then to predict the impact of these changes on the final system state (proportion of tasks in *Completed* or *Failed* states).

5 Method of Perturbation

The algorithm for incrementally perturbing selected rows is intended to predict broad trends rather than precise outcomes. It is a limited, brute-force search that is restricted in order to conserve resources, while exploring a reasonable range of alternatives. The algorithm permits simultaneous perturbation of combinations of two rows in order to capture situations where inter-row dependencies exist. In each perturbed row, each row element, corresponding to a column, with a probability of transition greater than zero is selected in turn for incremental increase. At the same time, the transition probabilities of one or more other row elements with non-zero values are decreased by a total equal to the increase, so that all elements in the perturbed row continue to sum to one. For a set of time-period matrices, these changes are applied to each matrix in the set. Each combination of altered transition probabilities represents a different execution path that the system may take.

The algorithm requires that a user first select a *primary row*, *r*, to perturb. The *secondary rows, s,* to be perturbed are then automatically determined, as described below. The user also must select a *perturbation limit* $\boldsymbol{L}$, on how far transition probabilities can be perturbed and also select the incremental amounts by which primary and secondary rows will be perturbed. These decisions define the extent and granularity of the perturbation that will take place. An overview of the procedure is provided below. For more detail, see (Dabrowski and Hunt 2008). This section also discusses the computational effort required to apply the perturbation algorithm to the Markov chain simulation. This effort is a small fraction of what would be necessary to explore the same set of alternative behaviours using the large-scale simulation. The next section, Section 6, then presents results of applying the perturbation algorithm and compares these results with those produced by the large-scale simulation.

5.1 Overview of Perturbation Algorithm

In the primary row, starting from numerically lowest row element, each element having a positive transition probability is used in turn to determine as the *primary increase column*, $c^{\uparrow}$. In this column, the transition probability is raised by a gradually increasing amount, m_{prim} up to the limit $\boldsymbol{L}$. These increases occur in increments defined by a *primary increase amount*, v_{prim}. At the same time, the other elements in the primary row are reduced by proportions of m_{prim} determined by weight factors, as follows. Each *non-increase column* in turn is selected as the primary column to decrement, termed a *primary sink column*, $c^{\downarrow}$. For the primary sink column, a *sink weight, w,* is selected from a predetermined set of sink weights called the *sinkWeightSet*. In the experiments reported here, the sinkWeightSet consisted of {0.2, 0.4, 0.6, 0.8, 1.0}. The probability of transition for $c^{\downarrow}$ is reduced by the amount $w \cdot m_{prim}$. The remainder of the weighted reduction, or $(1.0 - w) \cdot m_{prim}$, is distributed to the other *non-sink columns*. A perturbation of primary row *r* may be summarized by

$$p_{rj}^{(new)} = \begin{cases} p_{rj}^{(old)} + m_{prim} & j = c^{\uparrow} \\ \left[p_{rj}^{(old)} - w \cdot m_{prim} \right]^{+} & j = c^{\downarrow} \\ \left[p_{rj}^{(old)} - (1-w) \cdot m_{prim} \dfrac{p_{rj}^{(old)}}{\sum_{k \neq c^{\uparrow}, c^{\downarrow}} p_{rk}} \right]^{+} & j \neq c^{\uparrow}, c^{\downarrow} \end{cases} \quad (6)$$

for $p^{(old)}_{rj} > 0$ where $[a]^{+}=a$ if $a>0$ and $[a]^{+}=0$ otherwise. Of course, if *w* is 1.0, or if the primary increase column $c^{\uparrow}$ and primary sink column $c^{\downarrow}$ are the only columns with non-zero transition probabilities, the primary sink column bears the entire reduction.

The secondary row *s* can be selected on the basis of either: (a) the numeric value of the primary increase column $c^{\uparrow}$, if it is not equal to the number of the primary row, or $c^{\uparrow} \neq s$ (otherwise no secondary row is selected); and (b) by strength of association, using the total value of transition probabilities between the two states the rows represent and (if known) the number of transitions that occur between these states. The default method is (a); and this was used for the results reported below. Thus in the primary row *r*, as each primary increase column, $c^{\uparrow}$, is selected, a different secondary row is also selected. As in the primary row *r*, each positive row element in the secondary row, *s,* is selected in turn for increase, and the corresponding column is designated as the *secondary increase column*, $d^{\uparrow}$. However, in the secondary row, the perturbation is simpler--the transition probability of a secondary increase column, $d^{\uparrow}$, is raised by a *secondary increment amount*, v_{sec}, in 5 equal steps to produce successive perturbation amounts, m_{sec}, up to $\boldsymbol{L}$. As in the primary row *r*, transition probabilities in the remaining columns of the secondary row are decreased by an equal amount; though here the amount of decrease for each column is assigned in proportion to the relative value of its transition probability (similar to non-sink columns in the primary row). To summarize,

$$p_{sj}^{(new)} = \begin{cases} p_{sj}^{(old)} + m_{\text{sec}} & j = d^{\uparrow} \\ \left[p_{sj}^{(old)} - m_{\text{sec}} \dfrac{p_{sj}^{(old)}}{\sum_{k \neq d^{\uparrow}} p_{sk}} \right]^{+} & j \neq d^{\uparrow} \end{cases} \quad (7)$$

Each combination of variable assignments for the primary row, primary increase column, primary sink column, sink weight, together with the secondary row, secondary increase column, and secondary increase amount (if any) is considered a unique perturbation combination, labelled $\{r, c^{\uparrow}, c^{\downarrow}, w, s, d^{\uparrow}, m_{sec}\}$. For each perturbation combination, a separate perturbation

sequence of L/v_{prim} steps is carried out in the primary row. In each element of the perturbation sequence, the value of the primary increase column, $c^{\uparrow}$, is successively raised by v_{prim} while the primary sink column, $c^{\downarrow}$, and non-sink columns, if any, are decremented as described above. For a set of time period matrices, this perturbation sequence is applied to each matrix in the set. For each assignment of incremental values in the perturbation sequence, the Markov chain simulation procedure described in section 4.4 is carried out for the time-period matrix set. Each execution of the simulation represents a potentially different system execution path. The incremental increases in the perturbation sequence continue until the transition probability in the primary increase column, c↑, reaches $\boldsymbol{L}$ or 1.0 in each time-period matrix. Thus if L=0.25 and v_{prim} = 0.01, there are a maximum of 25 Markov chain simulations in a perturbation sequence for a perturbation combination.

5.2 Implementing the Perturbation Approach

The perturbation method described above was used to predict the result of failing to fulfil the three service guarantees described in section 3. To do this, we used the time-period matrix set in the Appendix (summarized by the weight-averaged matrix in Figure 2), together with the default method (a) for secondary row selection. The sink weight set and parameters values for $\boldsymbol{L}$, v_{prim}, and v_{sec} described above were also used. Applying the perturbation method resulted in generation of 2805 perturbation combinations and perturbation sequences consisting of 89,750 simulations, which required 3354 s (56 minutes) of computation time using an Intel Xeon MP processor. This is a substantial amount, but less than 0.5% of the time (205 hours) that the large-scale simulation needed to show behaviours described below in which the service guarantees were violated.

6 Comparing Perturbations of the Markov Chain and Large-Scale Simulation

The systematic perturbation of the TPMs described in the preceding section revealed a wide range of behaviours. A subset of these behaviours, corresponding to a subset of the total perturbation combinations discussed above, show what might occur if individual guarantees were violated. These perturbation combinations represent service guarantee violation scenarios of interest. In the Markov chain model, violation of the Discovery Guarantee corresponded to a subset of perturbation combinations for rows 1-4 in which tasks were prevented from transitioning to the *Discovering* state. Two of these combinations are presented here.

Failure to fulfil the Service Engagement Guarantee was enacted by reducing probability of transition from the *Negotiating* state to the *Monitoring* state in row 4. Perturbation combinations that reduced this probability represented a violation scenario in which SLAs were not granted even though users and providers might be eligible for, and should be able to obtain, agreements. Violation of the Service Fulfilment Guarantee was enacted by reducing the probability of transition from the *Monitoring* state to the *Completed* state in row 5, while increasing the probability of transitioning from *Monitoring* to another state. This violation scenario corresponded to aborting a task that was either executing, or in a waiting queue. The results of these perturbations of the Markov chain are shown in graphs of perturbation sequences for relevant perturbation combinations.

To compare the results of the perturbations to the Markov chain with similar changes to the behaviour of the large-scale simulation, the original model (Mills and Dabrowski 2008) was altered to simulate the effects of not fulfilling the three service guarantees. These changes to the large-scale model are described below and their effect on performance is graphed. In what follows, the results of perturbing both the Markov chain and large-scale simulation to emulate violation of the three service guarantees are described. These results are compared in terms of how well the Markov chain simulation predicts the result of the large-scale simulation and the relative computational effort required by each method.

6.1 Service Discovery Guarantee

The effect of not fulfilling the Discovery guarantee is shown in two selected violation scenarios in which initiation of discovery actions is prevented. In the first, the probability of transition from the *Initial* State to the *Discovering* state is lowered (row 1). In the second, the transition probability from the *Waiting* state to the *Discovering* state is lowered (row 2).

6.1.1 Perturbation of Transition to the Discovery state in Row 1

Row 1, column 3 of the unperturbed weight-averaged matrix shows the probability of transition from the *Initial* state to the *Discovering* state. The five-period matrix set in the Appendix shows this transition occurs entirely in the first time period of the simulated day. The transition from *Initial* to *Discovering* marks the arrival of a task in the grid system and is followed immediately by an attempt to discover providers to execute the task.

Figure 4 shows the effect of systematically lowering the probability of the transition from *Initial* to *Discovering* in the Markov chain simulation and the equivalent operation in the large-scale simulation. To perturb the Markov chain simulation, column 1 of row 1 is selected as the primary increase column, while column 3, *Discovering*, is designated as the primary sink column. Since there are no other columns in row 1 that have transition probabilities greater than 0, the sink weight is 1.0. Using the default secondary column selection method, no secondary row is perturbed, since the number of the primary row and increase column are the same.

In the large-scale simulation, the equivalent of reducing the probability of transition to the *Discovering* state was achieved by systematically increasing the amount of time each task remains in the *Initial* state, thus in effect delaying arrival of tasks into the grid system. This perturbation had the effect of *right-shifting* the arrival time distribution described above (Mills and Dabrowski 2008) and caused tasks to fail to meet their deadlines. When column 1 of row 1 was selected as the primary increase column in the Markov chain simulation, the same right shift was simulated, because recall that the Markov chain process was modified to allow task arrival

to take place using the distribution derived from the large-scale simulation (Section 4.1). Right-shifting this distribution delayed transition from the *Initial* to *Discovering* state, producing the same result.

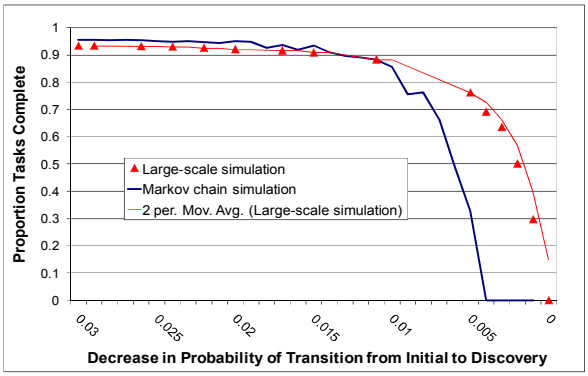

Figure 4. Proportion tasks complete in large-scale and Markov chain simulations in response to reduction in the probability of transition from *Initial* to *Discovering* (column 3 of row 1 is the primary sink column), while raising self-transition of *Initial* (column 1 is the primary increase column).

In Figure 4, curves for both large-scale and Markov chain simulations show that proportion of tasks complete decreases relatively little as the probability of transition from *Initial* to *Discovering* state is reduced from 0.03 to 0.01. Below 0.01, the proportion complete drops sharply in both cases. This reflects the effect of increasing delay of tasks leaving the *Initial* state to progress through the *Discovering*, *Negotiating*, and *Monitoring* states, so that they do not have sufficient time to execute and reach *Completed*. Although the Markov chain curve declines more steeply, it is similar to the curve for the large-scale simulation. Both show that performance will decline little until the probability of transition to *Discovering* falls below 0.01. In our experiments, the Markov chain simulation completely perturbed row 1 in 82.39 s, while, the large-scale simulation required 21.18 hours to capture the similar guarantee violation behaviour.

6.1.2 Perturbation of Transition to the Discovery state in Row 2

The probability of transition from the *Waiting* state to the *Discovering* state is shown in row 2, column 3. This transition represents subsequent attempts to initiate discovery operations by task clients after the first round of discovery which occurs when a task first arrives. In the Markov chain simulation, a violation scenario for the Discovery Guarantee was created by reducing the probability of transition from *Waiting* to *Discovering* (selecting column 3 as the primary sink column with a primary sink weight of 1) while raising the probability of transition from *Waiting* to *Negotiating* (selecting column 4 as the primary increase column). In this case, row 4 (*Negotiating*) was perturbed as the secondary row since the *Negotiating* state corresponds to column 4. Equivalent behavioural changes were made to the large-scale simulation by altering the code to prevent the task client from initiating subsequent rounds of discovery. The large-scale model was iteratively executed. On each iteration, the probability of delaying the start of a subsequent discovery phase was incrementally raised. TPMs were generated for this perturbed behaviour and compared with the Markov chain process.

Figure 5 shows the result of these alterations to the large-scale simulation together with the Markov chain perturbation sequences for 15 perturbation combinations of row 2 that best captured this violation scenario. Exploring all 425 perturbation combinations for row 2 required 373.44 s of computational time, while the large-scale simulation required 12.17 hours to execute the perturbed behaviour.

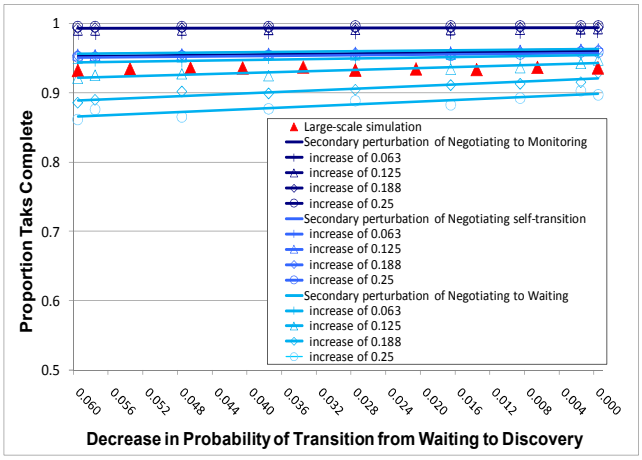

Figure 5. Proportion tasks complete in the large-scale and Markov chain simulations (shown using linear trend lines) in response to reduction in the probability of transition from *Waiting* to *Discovering* (column 3 of row 2 is the primary sink column) while raising the transition probability of *Waiting* to *Negotiating* (column 4 is the primary increase column).

In Figure 5, both the curves for the Markov chain and large-scale simulation show essentially no reduction in task completion as the probability of initiating subsequent discovery actions goes to zero. In the large-scale simulation, failing to initiate subsequent discovery does not affect task completion, because the discovery process is sufficiently efficient so that all eligible providers are found on the first discovery attempt (see Section 6.1.1). Hence, subsequent discovery actions are not actually needed, and the absence of these actions does not impact performance. The curves for the related Markov chain perturbation combinations shown in Figure 5 agree well with the large-scale simulation. If the Markov chain curves were used to make predications, they would accurately predict the result of the large-scale simulation with relatively minor differences in value of tasks completed. Other perturbations of transitions to the *Discovering* state in rows 3 and 4 describe violation scenarios of the Discovery Guarantee that also agree with the large-scale simulation. These are omitted due to lack of space. (Please see Dabrowski and Hunt 2008).

6.2 Service Engagement Guarantee

The act of engaging a service to execute a task is represented in the Markov chain by the transition from the *Negotiating* state to the *Monitoring* state. In row 4 of the TPM, the effects of non-fulfilment of the Service Engagement Guarantee can be illustrated by reducing this

probability of transition. This perturbation is meant to predict the effect of reducing acceptance of agreements because either users or providers fail to conclude SLAs they should enter into. Along with decreasing the probability of transition from *Negotiating* to *Monitoring* (making column 5 of row 4 the sink column), the time-period TPM set is perturbed by increasing the probability of transition from *Negotiating* to either *Waiting*, *Discovering*, or *Negotiating*, i.e., choosing columns 2, 3, or 4 of row 3 as the primary increase columns. Choosing column 2 or 3 involves selecting corresponding rows (rows 2 or 3) for secondary perturbation.

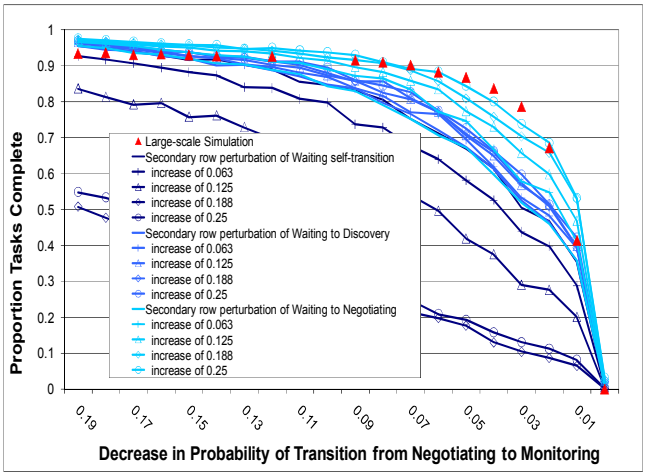

Figure 6. Proportion of tasks complete in the large-scale and Markov chain simulations in response to reducing the probability of transition from *Negotiating* to *Monitoring* (column 5 of row 4 is the primary sink column) while the transition probability to *Waiting* increases (column 2 is the primary increase column).

Figure 6 shows curves for perturbation sequences of 15 relevant perturbation combinations for row 4 (out the 785 total) in which the probability of transition from *Negotiating* to *Waiting* is raised (column 2 is the primary increase column) while the transition probability from *Negotiating* to *Monitoring* is lowered (column 5 is the *sink*, with a sink weight of 1). Row 2, *Waiting*, is chosen for secondary perturbation. In the large-scale simulation, the equivalent perturbation was accomplished by systematically increasing the probability that a provider rejects the agreement, the result of which is also shown in Figure 6. This figure shows that the perturbation of the Markov chain simulation is generally predictive of the large-scale simulation result. The Markov chain curves correctly predict that as the probability of transition to *Monitoring* falls to zero, i.e., users and providers fail to conclude SLAs, the proportion of tasks completed also falls to zero. In the case where secondary perturbation of row 2 increased *Waiting* self-transition to simulate additional delay, the Markov chain curves show that system performance would degrade still further.

A similar set of curves can be produced by the Markov chain simulation for perturbation combinations where column 3 (*Discovering*) is made the primary increase column while column 5 (*Monitoring*) remains the sink column. In this case, the Markov chain is generally predictive as well. This analysis is omitted due to lack of space (see Dabrowski and Hunt 2008). The computational cost for the entire 785 perturbation combinations required to perturb all of row 4 was 789.17 s. Raising ***L*** to 0.5 to obtain additional perturbation sequences needed to increase range in this case required another 1480.46 s. Here, the large-scale simulation required 41.57 hours.

6.3 Agreement Fulfilment Guarantee

The *Monitoring* state is entered once an SLA is concluded. In the Markov chain, failure to fulfil an agreement may be modelled by increasing the probability of transition from *Monitoring* to states other than the *Completed* state; namely, either increasing the probability of transition to the *Negotiating* state (representing a task abort) or increasing the probability of self-transition in the *Monitoring* state (representing an extended delay). In the former violation scenario, a task that transitions from *Monitoring* to *Negotiating* (aborts) may recover from this setback by later obtaining another SLA, returning to the *Monitoring* state, and then completing. In this section, this violation scenario is simulated in the Markov chain by making *Negotiating*, column 4 of row 5, the primary increase column, while making *Completed*, column 6, the sink column. In the resulting perturbation combinations, secondary row perturbation is applied to the *Negotiating* row (row 4). In the large-scale simulation, the equivalent behaviour change was enabled by systematically increasing the rate at which a provider aborts a queued or executing task. As before, the large-scale simulation iterated, with the abort rate increasing on each iteration.

Figure 7 shows the resulting curves for Markov chain perturbation sequences in which the probability of transition to the *Negotiating* state is raised (i.e., column 4 is the primary increase column) as the probability of transition to *Completed* (i.e., column 5 is the sink) falls from a weighted average of 0.008 to zero. The figure shows that this perturbation causes the proportion of tasks completed to fall dramatically. This figure shows 20 of the most relevant perturbation combinations (out of a total of 270) for the case where the *Completed* state has a sink weight of 0.2. Alternative perturbations using *Monitoring* as the sink column (not shown) produce a similar result. In all cases, the Markov chain curves show a pronounced reduction in proportion of tasks completed that is substantially predictive of the curve for the large-scale simulation, also shown in Figure 7. The figure shows that 5 of the 20 curves represent accurate approximations of the large-scale simulation result. The remainder show the distinct downward trend in tasks completed, though at markedly different slopes. In the 5 curves closest to the large-scale simulation result, the secondary row, *Negotiating* (row 4), is perturbed to raise the probability of transition to the *Monitoring* state (column 5). This effectively models a situation where tasks that fail to transition from *Monitoring* to *Completed* (abort) are later able to negotiate new SLAs, return to the *Monitoring* state, and complete—as might be expected in the real world. While only a relatively small proportion of the curves produced through Markov chain simulation are this accurate, all curves (including those not shown) predict that if the probability of transition from *Monitoring* to *Negotiating* is raised sufficiently (i.e., the probability of task abort is high enough), system performance will drastically degrade. The Markov chain

simulation required 260.2 s of execution time to process the 270 perturbation combinations for row 5. The large-scale simulation required a lengthy 122.6 hours to carry out the equivalent perturbation behaviour, because repeated task aborts entailed extensive delays.

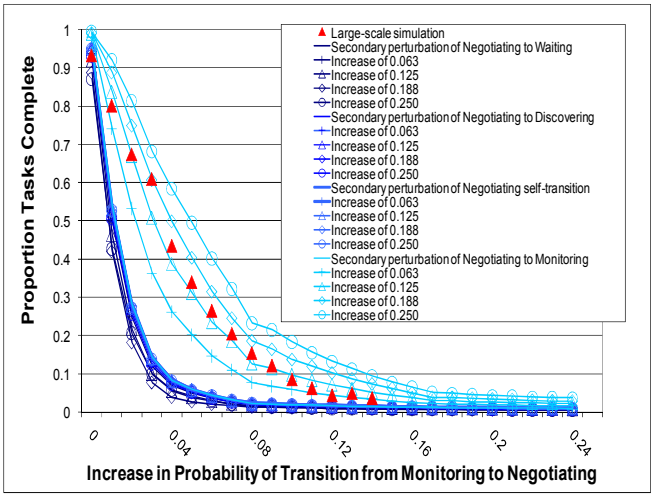

Figure 7. Proportion tasks complete in large-scale and Markov chain simulation in response to increasing the probability of transition from *Monitoring* to *Negotiating* (column 4 in row 5 is the primary increase column) while decreasing the probability of transition from *Monitoring* to *Completed* (column 6 is the sink column with a sink weight of 0.2) from 0.008 to 0.

6.4 Summary of Analysis and Outstanding Issues for Future Work

The preceding sections showed that perturbation of TPMs carried out through application of the method described in sections 4 and 5 does indeed generally predict how the large-scale grid system will perform when key service guarantees are violated. In the case of the Discovery guarantee, the Markov chain was found to accurately correspond to the large-scale simulation in both cases shown. For the Engagement guarantee, perturbations of the Markov chain were found to generally correspond to the results produced by the large-scale simulation for one violation scenario. For the violation of the Service Fulfilment Guarantee, the perturbation of the Markov chain accurately captured the behaviour exhibited by the large-scale simulation. Thus perturbation of the Markov chain was show to be an effective predictor for all cases shown in this paper (for additional examples, see Dabrowski and Hunt 2008). In no case, did the Markov chain simulation produce results that contradicted the large-scale simulation. Moreover, the Markov chain approach achieved these results at less than 0.5% of the computational cost of the large-scale simulation. If the required data was obtained from a real-world system to create a Markov model and related TPMs, it is reasonable to believe that comparable results could be achieved.

Despite this success, important issues still remain to be resolved. The most important is scalability, which has three aspects. First is whether the approach scales with respect to the size of the system being modelled, as expressed in terms of such variables as number of entities being modelled, number of transitions taken, and workload. As section 4 has shown, the method of counting state transitions and generating transition probabilities is a straightforward arithmetic process that clearly does not depend on number of transitions. Here, scale does not hinder analysis. Second, there is the all-important issue of the size of the state model, that is, the number of states and the corresponding size of the TPM. Here, further work incorporating lumping techniques described in Section 2 will be needed. Finally, it is important to consider scalability with respect to the number of perturbations, or alternative execution paths, investigated. Despite the dramatic reduction in execution time seen for Markov chain method (< 0.5% of the execution time used by large-scale simulation), scalability may not be good for very large matrices or if many perturbations are needed. Follow-on research will be needed to examine this issue. Here, there is the possibility of extending non-linear algebra techniques and matrix methods (Stewart and Sun 1990) to generate eigensystems that can be analysed to determine what parts of the matrix are most sensitive to perturbation and thus where investigation should be focused. Despite these issues, the Markov chain approach entails dramatically less computational effort than large-scale simulation. Beyond scale, other issues exist, such as improving the accuracy of Markov chain simulation by selecting optimal sizes for time periods and identifying appropriate tests of statistical significance to measure accuracy.

7 Conclusions

Section 6 showed that perturbation of TPMs and Markov chain simulation was generally predictive of changes to performance arising from failure to fulfil basic service guarantees provided by grid computing systems. While Markov chain analysis did not reproduce the exact performance curves generated by the large-scale simulation, a carefully limited brute-force perturbation of TPMs produced a family of related curves which estimated the impact of not fulfilling service guarantees. Perturbed TPMs produced by Markov chain analysis were predictive both in cases where changes to the behaviour of the large-scale simulation resulted in severe performance degradation as well as cases where changes to the large-scale simulation did not significantly impact results. Thus, it is possible to conclude that the approach to perturbing Markov chains described in this paper did indeed answer the questions posed in section 3; namely, how non-fulfilment of the three service guarantees affects performance of a large-scale grid system. Moreover, the Markov chain procedure performed the analysis needed to answer this question using only 0.5% of the computational resources (less than two orders of magnitude) that was needed for the large-scale simulation. If, instead of the large-scale simulation, a real-world system could be used as a testbed in which conditions are sufficiently controlled to allow execution of repeated trials, the contrast in time (and resource) expenditure could be much greater. The study thus shows that Markov chain analysis is a valuable tool for understanding complex system behaviour in large-scale grid systems and can be used to predict performance changes that result when fundamental guarantees of service are not met.

8 References

Akioka, S. and Muraoka, Y. (2003): The Markov Model Based Algorithm to Predict Networking Load on the Computational Grid. *Journal of Mathematical Modelling and Algorithms* **2:** 251–261.

Andrieux A., Czajkowskim K., Dan, A., Keakey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S. and Xu, M. (2007): *Web Services Agreement Specification (WS-Agreement)*. GFD.107, Open Grid Forum.

Aupperle, B. and Meyer, J. (1991): State space generation for degradable multiprocessor systems. *Twenty-First International Symposium on Fault-Tolerant Computing (FTCS-21), Digest of Papers*, Montreal Canada, 308-315, IEEE Computer Society Press.

Balakrishnan, M. and Reibman, A. (1994): Reliability models for fault-tolerant private network applications. *IEEE Transactions on Computers*. **43** (9): 1039-1053.

Buchholz, P. (1995): Hierarchical Markovian Models: Symmetries and Reduction. *Performance Evaluation* **22** (1): 93-100.

Cao, X. (2005): Basic Ideas for Event-Based Optimization of Markov Systems. *Discrete Event Dynamic Systems: Theory and Applications* **15** (2): 169–197.

Cao, X. and Zhang, J. (2008): Event-Based Optimization of Markov Systems. *IEEE Transactions on Automatic* Control **53** (4): 1076-1082.

Carr D., How Google Works. *Baseline Magazine*, July 6, 2006. http://www.baselinemag.com. Accessed 15 July 2008.

Cassandras, C., Lee, J. and Ho, Y. (1990): Efficient parametric analysis of performance measures for communication networks. *IEEE Journal on Selected Areas in Communications* **8** (9): 1709-1722.

Chun, B., and Culler, E. (2002): User-centric performance analysis of market-based cluster batch schedulers. *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, Berlin Germany, 30, IEEE Computer Society Press.

Chiola, G., Dutheillet, C., Franceschinis, G. and Haddad, S. (1993): Stochastic Well-Formed Colored Nets and Symmetric Modeling Applications. *IEEE Transactions on Computers* **42** (11): 1343-1360.

Dabrowski, C. and Hunt F. (2008): *Markov Chain Analysis for Large-Scale Grid Systems.* Draft NIST Internal Report.

Goseva-Popstojanova, K., and Trivedi, K. (2001): Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*. **45** (2-3): 179-204.

Ho, Y. (1985): A Survey of the Perturbation Analysis of Discrete Event Dynamic Systems. *Annals of Operations Research* **3** (8): 393-402.

Ho, Y. and Li, S. (1988): Extensions of infinitesimal perturbation analysis. *IEEE Transactions on Automation Control* **AC-33** (5): 427-438.

Jacobi, M. and Gornerup, O. (2007): A Dual Eigenvector Condition for Strong Lumpability of Markov Chains. arXiv.org, http://arxiv.org/abs/0710.1986. Accessed 20 November, 2008.

Kemeny, J. and Snell, J. (1976): *Finite Markov Chains*. New York, Springer.

Laprie, J. and Kanoun, K. (1992): X-ware reliability and availability modeling. *IEEE Transactions on Software Engineering* **18** (2): 130-147.

Li, J., Blumenfeld, D., Huang, N. and Alden, J. (2008): Throughput analysis of production systems: recent advances and future topics. *International Journal of Production Research*. To appear,

Mills K. and Dabrowski C. (2006): Investigating Global Behavior in Computing Grids. *Lecture Notes in Computer Science* **4124**: 120-136, Springer.

Mills, K. and Dabrowski C. (2008) Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace? *Journal of Grid Computing* **6** (3): 291-311.

Nicol, D., Sanders, W. and Trivedi, K. (2004): Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing* **1** (1): 48 – 65.

Obal, W. and Sanders, W. (2001): Measure-adaptive state-space construction. *Performance Evaluation* **44** (1-4): 237-258.

Raffo D. (2006): Grid, redundancy, and home-cooked management help site survive. *Byte and Switch*, November 22, 2006.

Rosenberg, D., Solan, E. and Vielle N. (2004): Approximating a Sequence of Observations by a Simple Process. *The Annals of Statistics* **32** (6): 2742-2775.

Sanders, W. and Meyer, J. (1991): Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications, special issue on Computer-Aided Modeling Analysis, and Design of Communication Networks* **9** (1): 25–36.

Schweitzer, P. (1968): Perturbation Theory and Finite Markov Chains. *Journal of Applied Probability* **5** (2): 401-413.

Siegle, M. (1992): On Efficient Markovian Modelling. *Proceedings of the QMIPS Workshop on Stochastic Petri Nets*, Sophia Antipolis, France, 213-225.

Song, B., Ernemann, C. and Yahyapour, R. (2004) Parallel Computer Workload Modeling with Markov Chains. *Lecture Notes in Computer Science* **3277**: 47-62, Springer.

Stewart, G. and Sun, J. (1990) *Matrix Perturbation Theory*. San Diego USA, Academic Press.

Suri, R. (1989): Perturbation Analysis: The State of the Art and Research Issues Explained via the GI/G/l Queue. *Proceedings of the IEEE* **77** (1): 114-138.

Trivedi, K., Ramani, S. and Fricks, R. (2003): Recent advances in modeling response-time distributions in real-time systems. *Proceedings of the IEEE* **91** (7): 1023-1037.

Yeo, C. and Buyya, R. (2005): Service level agreement based allocation of cluster resources: handling penalty to enhance utility. *Proceedings of the 7th IEEE International Conference on Cluster Computing*. Boston, USA, 27-30, IEEE Computer Society Press.

Zakarian, A. and Kusiak, A. (1997): Modeling manufacturing dependability. *IEEE Transactions on Robotics and Automation* **13** (2): 161-168.

9 APPENDIX Five Time-Period Transition Matrices Showing Non-Homogeneity With Respect to Time

Time Period 1, 0-7200 s

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0.9697	0	0.0303	0	0	0	0
Waiting	0	0.8072	0.0550	0.1378	0	0	0
Disc	0	0.0631	0.6013	0.3356	0	0	0
Ngt	0	0.3506	0.0132	0.2920	0.3442	0	0
Mon	0	0	0	0	0.9961	0.0039	0
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	0

Time Period 2, 7201-14400 s

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0	0	0	0	0	0	0
Waiting	0	0.8527	0.0722	0.0749	0	0	0.0002
Disc	0	0	0.4988	0.5012	0	0	0
Ngt	0	0.5466	0.0316	0.3770	0.0448	0	0
Mon	0	0	0	0.0001	0.9928	0.0071	0
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	1.0

Time Period 3, 14401-21600s

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0	0	0	0	0	0	0
Waiting	0	0.8494	0.0730	0.0702	0	0	0.0074
Disc	0	0	0.6407	0.3593	0	0	0
Ngt	0	0.7845	0.0143	0.1854	0.0157	0	0.0001
Mon	0	0	0	0.0010	0.9828	0.0162	0
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	1.0

Time Period 4, 21601-28800s

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0	0	0	0	0	0	0
Waiting	0	0.8340	0.0722	0.0689	0	0	0.0249
Disc	0	0	0.9991	0.0009	0	0	0
Ngt	0	0.9899	0	0	0	0	0.0101
Mon	0	0	0	0.0013	0.9317	0.067	0
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	1.0

Time Period 5, 28801-36000s

	Initial	Wait	Disc	Ngt	Mon	Comp	Fail
Initial	0	0	0	0	0	0	0
Waiting	0	0.8111	0.0720	0.0734	0	0	0.0435
Disc	0	0	0.9991	0.0009			
Ngt	0	0.9899	0	0	0	0	0.0101
Mon	0	0	0	0.0013	0.9317	0.0670	
Comp	0	0	0	0	0	1.0	0
Fail	0	0	0	0	0	0	1.0