Alleviating the Impact of Churn in Dynamic Grids

K. Abdelkader, and J. Broeckhove

***Abstract*—The utilization of desktop grid computing in large-scale computational applications is an important issue at present for solving compute-intensive problems. However, such large-scale distributed systems are subject to churn, i.e., continuous hosts arrival, leaving and failure. In this paper we address the problem of churn in dynamic grids, and evaluate the impact of reliability-aware resource allocation on the performance of the system.**

***Index Terms*—auctions market, grids, grid economics, resource management, spot markets.**

I. Introduction

THE utilization of desktop grid computing in large-scale computational applications is an important issue at present. Platforms such as BOINC [1], [2] and SZTAKI [3], [4] that used to provide large-scale intensive computing capability have attracted recent research interest. Such platforms are referred to as volunteer grids or public resource grids [5], [2] where the hosts or providers are typically end-users' public PCs (e.g. homes, offices, universities or institutions) located at the edge of the Internet. Studies aimed at evaluating host availability have shown that providers connect to and disconnect from the grid without any prior notification. This effect which is called churn [6]. To optimize the performance of the system that is subject to churn, we shall consider the churn characteristics of providers i.e. the rate of connection/disconnection and the duration of the corresponding time periods. Also, it is significant from the perspective of the grid user, to consider the number of jobs failing and succeeding without resubmission being required [7]. These effects have to be taken into account when devising an appropriate allocation policy for such systems.

Our main contribution in this paper is to focus on how to reduce the effect of churn by reducing jobs failure due to downtime prior to job completion. We look for allocation strategies that are economically based, but that also take into account the apparent reliability of the providers. We aim to do this in a manner as simple as possible, by taking into account historical information of the providers and using this to screen the bids they submit to the market.

II. Modeling churn

In this paper the model adopted for the churn play a key role. Basically one models the distribution of the time durations during which a resource is available or unavailable. When the computing system is the resource available/unavailable translate to machine uptime/downtime. This is similar to the system-based churn model as described in [8]. One can also look at the availability of the CPU, which might differ, due to the provider policies or preferences, from the machine availability.

There exists quite a bit of literature on the subject. The authors in [9] have tackled the problem of modeling machine availability in enterprise-area and wide-area distributed computing settings. They have used three different data traces to measure resource availability, namely the UCSB SCIL [1] data set, the Condor [10], [11] data set and the Long-Muir-Golding data set. The results indicate that either a hyperexponential or Weibull model most effectively represents machine availability periods.

The authors in [12] focus on *CPU* availability. This differs from the previous work that looks at machine availability. Here one includes the case where, without the machine going down, the user or some monitoring daemon makes the CPU unavailable for grid work. The authors have measured and characterized the *time dynamics of availability* in large-scale Internet distributed systems with over 110,000 hosts. Their characterization has focused on identifying patterns of correlated availability. They instrumented the BOINC [1] client to record start and stop times CPU availability. The goal was to detect patterns of availability using K-mean algorithm [13] for clustering resources by their availability traces.

In [14], the authors studied the availability of resources, including CPU. They relied on trace data gathered from over 48,000 Internet hosts participating in SETI@home project under BOINC [2]. They attempted to show that a deployment of enterprise services in a pool of volatile resources is possible and incurs reasonable overhead.

The authors in [15] have analyzed the Failure Trace Archive (FTA) that comprises public availability traces of parallel and distributed systems. They inspected the basic statistics of the traces, and, they fit the distributions for modeling failures in terms of probability distributions of availability and unavailability intervals. They implemented a toolbox to facilitate the comparative analysis of failure traces. This toolbox was implemented in Matlab, enhanced with several open source Matlab packages. Using the toolbox, they made a uniform and global statistical analysis of failure in nine distributed systems. One key finding was that the Weibull and Gamma distributions are often the best candidates for availability and unavailability distributions.

In this work we consider provider availability, a binary value that indicates whether a provider is reachable and responsive. This corresponds to the definition of availability in [16], [17]. By contrast, the authors in [18], [19] addressed the problem of *CPU* availability instead of provider or host availability. Of course provider unavailability implies *CPU*

Manuscript received March 8, 2012; revised April 01, 2012.

K. Abdelkader is with the Department of Computer Sciences, Higher Institute of Comprehensive Professions, Ghadames, Libya .

J. Broeckhove and K. Abdelkader are with the Department of Mathematics and Computer Sciences, University of Antwerp, Middelheimlaan 1, 2020, Antwerp, Belgium e-mail: (Jan.Broechove@ua.ac.be, Abdelkader.Khalid@gmail.com).

[1]At the University of California, Santa Barbara (UCSB) they gathered measurements of the time between machine reboots of the publically accessible workstations in the Computer Science Instructional Laboratory (CSIL)

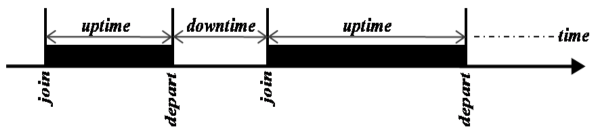

Fig. 1. Provider availability and unavailability time periods indicated in the *uptime* and *downtime* intervals.

unavailability, but the converse is not true, particularly when multi-processor and or multi-core machine are involved. The state transition of provider availability to unavailability and back is depicted in figure 1 that represents a timeline with the following time intervals:-

- ***uptime*** stage: a provider is available and when a job is allocated to the provider it uses all the CPU power of that provider
- ***downtime*** stage: a provider has withdrawn from the grid system because of policy decision, shutdown, …etc

Churn is modeled with two provider-level characterizations. Firstly, the *uptime* length distribution, which is one of the most basic properties of churn. It captures the period of time that the providers remain active in the grid system each time they appear. Secondly, the *downtime* can be defined as the interval between the moment a provider departs from the grid system and the moment of its next arrival in the system (see Figure 1). Most churn studies use these two distributions to model a churn. A provider's *lifetime* is the time between the moment that a provider first participates in the grid system and the time that a provider finally and permanently exits from the grid system.

We will in the remainder of this paper refer to the provider's reliability over an interval of elapsed time $[t_a, t_b]$. Given uptime periods $uptime(k)$ for $k = 0, 1, 2, ...$ starting respectively at time t_k^i and ending at t_k^f, the reliability is defined as the aggregate of the uptime periods within that time interval, divided by the elapsed time

$$R = \frac{\sum_k \max(t_b, t_k^f) - \min(t_a, t_k^i)}{t_b - t_a} \qquad (1)$$

The reliability index R thus represents the fraction of the elapsed time that the provider was available in the time interval from t_a to t_b. Obviously, the higher R the more available the provider. This is a narrow definition that equates reliability with provider availability.

III. GES MODEL

In this section we present the model and scenarios used in the Grid Economic Simulator to analyse methods for alleviating the effects of churn in dynamic grid systems.

The simulation model consists of three key elements. Firstly, a set of N geographically distributed grid providers "resource owners" denoted by $P_1, P_2, ..., P_N$, each of which is committed to deliver computational power. In addition, there is a group of potential providers, providers that are in a waiting state but that are ready to join and deliver computational resources in the grid.

Secondly, a market for resource allocation and job scheduling. All resource owners follow the same pricing strategy for determining the outcome of the bidding process in the market.

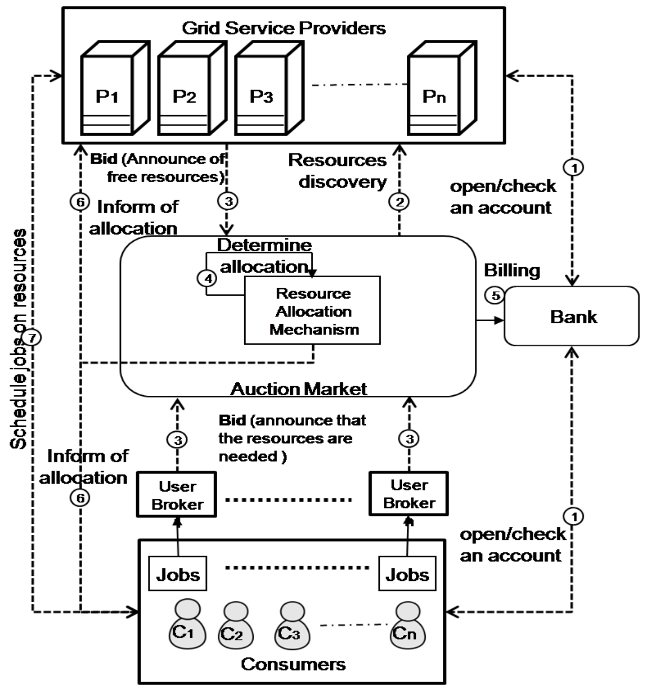

Fig. 2. An overview of the auction market architecture in GES.

Thirdly, M resource consumers or "users" denoted by $U_1, U_2, ..., U_M$, that are also geographically distributed and each has a queue of jobs to be executed. A job is specified by its job length and budget and is used to acquire resources for the job execution. In our simulation, the job lengths are randomly selected from a uniform distribution. The jobs are a CPU-bound computational task. The consumers are subdivided into four groups such that each has different deadlines for the jobs to be finished. That is, the jobs of each group have to be completed before the deadline of that particular group with the initial budget that has been allocated to the job.

The consumers interact with resource brokers that hide the complexities of grids by transforming user requirements into schedules for jobs on the appropriate computing resources. The brokers also manage the job submission and collect the results when they are finished. For instance, the consumer U_m, where $(m = 1, 2, ..., M)$ sends the job $J_{j,m}$ with its bidding value $b_{j,m}$ to the broker. In accordance to consumer's request, one of the available resource providers P_n, where $(n = 1, 2, ..., N)$ will receive this request. The broker plays a complex role of a wide range of tasks, i.e. it is an intermediary between resource providers and consumers. Beside this functionality, the broker provides information about the status of CPU usage in the grid system.

Finally, an entity that functions as a bank is used. When a job failure occurs, the broker in this case will send a report to the bank. The bank refund the money that had already been prepaid by U_m to the account number of P_n. This enables the consumer to recover the money and use it to resubmit the failed jobs.

In Figure 2 one finds a graphical representation of all the entities involved in the model and the key steps in the flow of control.

A. Decentralized marketplace

As mentioned previously, GES applies market-based principles for resource allocation to application scheduling. In this section we describe how resources are priced. In effect, we adopt an auction market for the pricing of computational resources. In contrast to the previous work [20], where the motivation was focused on price stability using a commodity market, the auction market has been engineered to be more realistic in geographically distributed systems.

The decentralized auction is a first-price sealed-bid (FPSB) auction with no reserve price (see the diagram 2), where the auctioneer collects all received bids and subsequently computes the outcome. The bidders can submit only one bid. The resource is allocated to the highest bidder at the end of the auction. Consumer U_m typically, has its own valuation $v(R_{i,n})$ for resource $R_{i,n}$ to bid. The consumer U_m communicates its willingness to pay $(b_{j,m})$ for resource $R_{i,n}$ and the required processing time for job, $J_{j,m}$. The resource information concerning resource $R_{i,n}$ of provider P_n consists of information about the CPU such as the $CPUspeed$, μ_i. The information on job J_j consists of the job length, l_j, the job deadline d_j and the budget B_j available for execution of the job.

When the auction ends by awarding the highest bid, the auctioneer charges the winner an amount of c_n (i.e. $c_n = (b_{j,m})$) per time step of the job for resource usage. That is, the charges are determined at allocation time and remain fixed throughout job execution. The required time for the $J_{j,m}$ to execute on $R_{i,n}$ and the associated cost are computed using the equations 2 and 3 respectively.

$$T(J_{j,m}, R_{i,n}) = \frac{l_{j,m}}{\mu_i} \tag{2}$$

$$B(J_{j,m}, R_{i,n}) = c_m * T(J_{j,m}, R_{i,n}) \tag{3}$$

B. The role of churn

When churn is incorporated in the above model, leading to a dynamic grid system, jobs may fail because the provider withdraws from the grid system. Also users can withdraw from the grid system. The churn itself is modeled through system uptimes and downtimes as explained in section II. As a matter of fact, we have only included provider churn in the model. In case a consumer withdraws from the market, their results do not get recovered by them and they will have to resubmit their jobs. There is however no impact on the functioning of resource allocation of the grid system as a whole.

When providers withdraw from the market, jobs execution fails. Consequently, consumers have to be reimbursed; they have to resubmit their jobs. In an effort to meet their deadlines the consumer may have to increase their bids. Thus it is necessary to try and submit the jobs to a provider that is not likely to fail. In the next section we propose such an approach.

C. Alleviating the role of churn

In order to alleviate the adverse effect of churn on the usefulness of the grid system we propose a simple, straightforward algorithm that only exploits the historical information on uptimes and downtimes. This information is maintained in the grid information system and is certainly not privileged information. The basic quantity that is used in the algorithm is the reliability index introduced previously (equation 1) for provider P_i in the time interval from t_a to t_b

$$R(P_i) = \frac{\sum_k \max(t_b, t_k^f) - \min(t_a, t_k^i)}{t_b - t_a} \tag{4}$$

where t_k^i and t_k^f are the start and end times respectively of the k^{th} uptime period of P_i in that time interval. The reliability index thus represents the fraction of the elapsed time that the provider was available in the time interval from t_a to t_b, and characterizes each provider individually. We also consider a threshold number which we denote γ and refer to as the `reliability threshold`. This threshold is the applied in a very straightforward manner: consumer will only bid with providers P_i such that

$$R(P_i) \geq \gamma \tag{5}$$

That is, providers whose reliability index exceeds gamma. This has the effect of screening the less reliable providers. It is of course a simple heuristic rule, based on information of the provider's track records. It does not involve any statistical evaluation nor does it evolve any quality of service elements. It does however lead to quantitative improvements of the job failures rates due to churn as we shall see from the simulation results in the next section.

IV. Simulation results

In this section, we evaluate the effect of churn in the above model through two simulation scenarios. The key parameters that apply in all scenarios discussed in this section are listed in table I. The relative workloads on the grid system that are assigned are roughly the same for those scenarios that we compare with and without churn. The relative workload, ϕ, can be defined as the ratio of aggregate workload, l, to the aggregate computational capacity, α.

$$\phi = \frac{\sum_m l_m}{\sum_n \alpha_n} \tag{6}$$

This relative workload ϕ highly affects the system performance. Because of this, scenarios where we directly compare the effects of churn by executing the scenario with and without churn have been designed to run under the same relative workloads.

In all experiments we have divided the users into four groups, each characterized by a different range of deadlines to be met for the jobs that the user needs to have executed (see table I). In some of the experiments we have also subdivided the providers into groups with different parameters for the uptime churn distribution. Simulation parameters that differ per scenario are reported in the appropriate subsection.

A. Performance metric

The performance objective for consumers is a high rate of successfully executed jobs within the deadline. Jobs may fail to meet this objective because of provider churn but also simply because the aggregate computational load on the grid system is such that the consumer fails to acquire

TABLE I
SIMULATION PARAMETERS.

Simulation steps	2000_s	
Number of consumers	6000	
Number of providers	1000	
Initial budget	1000000	
$d_i(Group1)$	$\{1, \cdots, 100\}$	
$d_i(Group2)$	$\{1, \cdots, 1300\}$	
$d_i(Group3)$	$\{1, \cdots, 1700\}$	
$d_i(Group4)$	$\{1, \cdots, 2000\}$	
microsoft99 Scenario	K	λ
uptime	0.55	35.30
downtime	0.60	9.34
pl05 Scenario	K	λ
uptime	0.33	19.35
downtime	0.36	5.59

TABLE II
SCENARIO I SIMULATION PARAMETERS WITH `pl05` DISTRIBUTION.

pl05 case	K	λ
Job duration in time steps	$\{100 \cdots, 110\}$	
Nr. of jobs per user at injection step	3	
uptime_1	0.33	14.51
uptime_2	0.33	19.35
uptime_3	0.33	24.19

the necessary computational resources within deadline and within their budget constraint. We are interested in studying the former effect.

In order to discriminate between both effects we define the *failure rate* (F_{rate}) with the following equations:

$$F_{rate} = \frac{FJ}{TJ} - F_{rate_{noChurn}} \tag{7}$$

$$FJ : \textit{number of jobs fail} \tag{8}$$

$$TJ : \textit{number of total jobs} \tag{9}$$

$$F_{rate_{noChurn}} : \textit{failure rate with no churn} \tag{10}$$

Thus, the *failure rate* represents the fraction of jobs that fail, but we do not count those that would fail even if there were no churn in the system. That is to say, the *failure rate* measures the job failures directly attributable to provider churn in the grid system.

B. Scenario I: The effect of the mean uptime

In this scenario we want to explore how is sensitive the evolution of the grid system, the providers, user, and executed jobs, with respect to parameters of the churn model. We have done a number of experiments in this respect and report here on one such experiment that is representative of this type of exploration.

We have the simulation with the parameters as in tables I and II. We use the `pl05` churn model with the parameter determining the uptime at three different values: $\lambda = 14.51, 19.35, 24.19$. The middle value is the actual `pl05` value, the other two are an increase and decrease by 25 percent respectively. An increase of λ represents an increase of the average provider uptime.

As the mean uptime increases or decreases for different λ the number of providers churned out of the grid system increases and decreases respectively. This inverse relationship confirms what one would expect a priori.

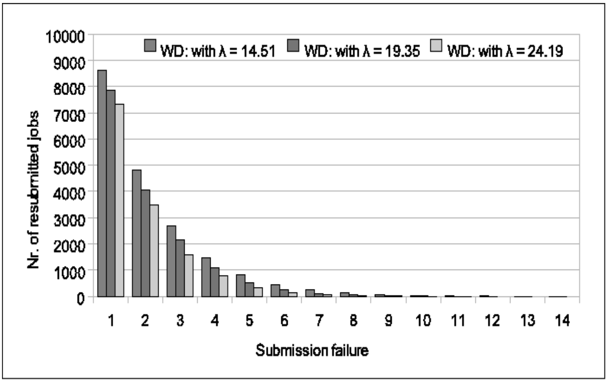

Fig. 3. Scenario I: the number of jobs resubmitted at least once, at least twice, etc. with churn of the `pl05` distribution.

Figure 3 indicates the number of jobs that need to be resubmitted because of failure at least once, at least twice, ... Again we show the results for simulations based on the three different λ parameters. The total number of jobs initially in the system for this case is 18000.

The results of Figure 3 lead to a number of observations. Apparently this system has a heavy workload and churn. Approximately one in two jobs needs to be resubmitted at least once, one in five at least twice. The tail of multiple jobs submissions is quite long.

The differences between the different λ cases is as one would expect: smaller λ means shorter uptimes and leads to more resubmissions, larger λ means longer uptimes and leads to fewer resubmissions. There is no noticeable effect on the multiplicity of the resubmissions.

TABLE III
SCENARIO II SIMULATION PARAMETERS.

microsoft99 case	
Nr. of jobs per user at injection step	7
Job duration in time steps	$\{40 \cdots, 50\}$
pl05 case	
Nr. of jobs per user at injection step	3
Job duration in time steps	$\{100 \cdots, 110\}$

C. Scenario II: The threshold algorithm

In this scenario, the experiments are effected using the parameters listed in tables I and II. In the experiments we explore the effect of applying the threshold algorithm. The differences in job duration between the `microsoft99` and `pl05` cases are related to the differences in the average uptime in the two churn distributions. They have been set to generate roughly identical relative workloads in the grid system in both cases.

For each provider, at every point in time, the reliability index is computed. Users the screen providers i.e. they submit bids for computational resources only with those providers whose reliability index exceeds a preset threshold value. We explore the effect of changing the threshold on the job failure rate observed in the grid system.

Figure 4 shows the failure rate as a function of the threshold, both for the `micrsoft99` and `pl05` churn distributions. For the latter the rate decreases slightly up to $\gamma = 70\%$ while the former hardly changes at all. Form $\gamma = 70\%$ on the failure rate steadily climbs.

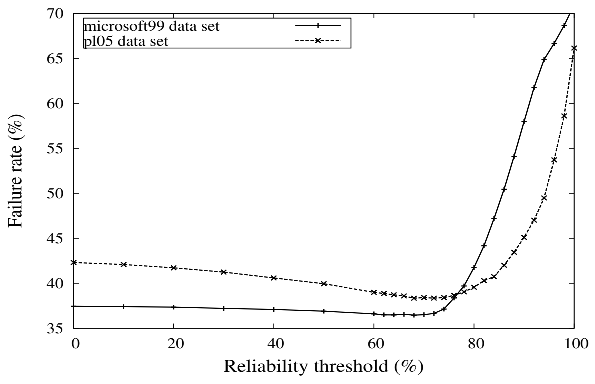

Fig. 4. Scenario II: the job failure rate for different thresholds γ.

The first observation is somewhat disappointing because it basically indicates that the threshold algorithm does not really improve the failure rate compared to the situation with $\gamma = 0\%$ i.e. when no reliability screening is applied. The second observation is rather easy to understand: when one increases γ, the pool of providers available for bidding shrinks. Thus, the higher γ the smaller the set of available providers. Even though these might be the more reliable ones, the computational capacity shrinks in such a manner these jobs simply to find computational resources to run on and fail to meet deadline.

In the panels in Figure 5 the results of the same experiments are shown, but now differentiated per user group. Four subgroups of users have been introduced that each differs in the deadlines that are set for their jobs. The relevant parameters are listed in see table I. The deadlines for *Group*1 are the most stringent, those of *Group*4 are the least stringent.

The results in these figures indicate that the failure rate in the `pl05` case is more responsive to application of threshold algorithm than that in the `microsoft99` case. This observation is consistent with the aggregate (over the user groups) failure rate shown in Figure 4.

The results also indicate that the improvement in the failure rate, even if slight, is most pronounced when the deadline is the least stringent and vice versa. We have no clear, unambiguous indications as to the cause of this relation because of the complexity of the system. However, we conjecture that the effect is most probably due to the fact that the more stringent the deadline, the less effective the mechanism of resubmissions and because of that also the less effective the screening of unreliable providers.

TABLE IV
SCENARIO IV SIMULATION PARAMETERS.

microsoft99 case	
Nr. of jobs per user for load_1	7
Nr. of jobs per user for load_2	4
Nr. of jobs per user for load_3	2
Job duration in time steps	$\{40 \cdots, 50\}$
pl05 case	
Nr. of jobs per user for load_1	3
Nr. of jobs per user for load_2	2
Nr. of jobs per user for load_3	1
Job duration in time steps	$\{100 \cdots, 110\}$

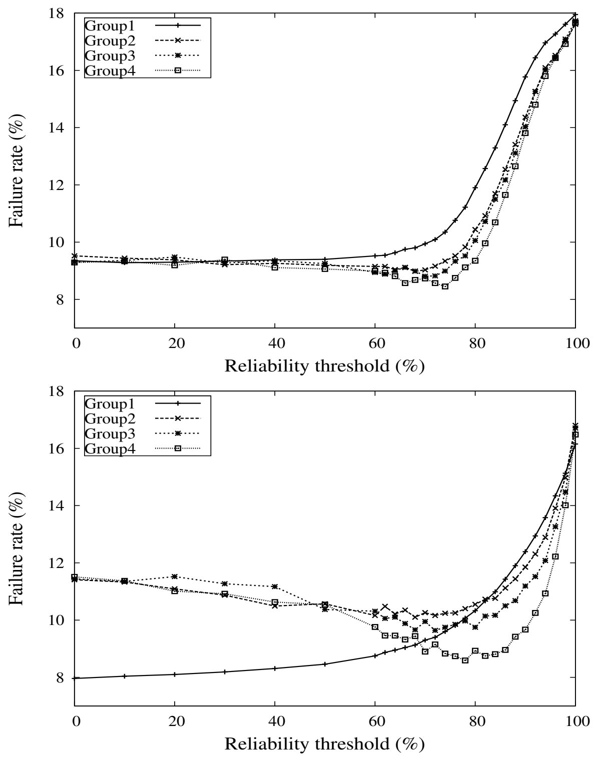

Fig. 5. Scenario II: job failure rates per consumer group for different γ for `microsoft99` (top panel) and `pl05` (bottom panel) distributions.

D. Scenario III: Relative workload

In this scenario we study the effect of the system workload on the failure rate. To do so, we run the same experiments under different workload conditions by changing the number of jobs per consumer (see table IV for a listing the simulation parameters).

Figure 6 shows the failure rates as a function of γ, the reliability threshold, for each of the workload conditions, both for the `microsoft99` (top panel) and `pl05` (lower panel) churn distributions. It is obvious here that the lighter the effective workload on the grid system, the more pronounced the improvement of the failure rate due to the threshold algorithm. This can be understood in the following sense. The application of the threshold criterion has two inherent, counteracting effects: the higher the threshold is set, the more reliable the providers satisfying the criterion but also the fewer the number of providers passing it. The first effect will lower the failure rate because providers are less likely to churn out and cause the jobs to fail. The second effect will increase the failure rate because there are fewer providers available, which lowers the total computational capacity in the grid and causes jobs not to finish before deadline or not be run at all. At a certain threshold value γ an optimum equilibrium between these opposing effects is achieved and the failure rate is at its lowest value, given the other parameters of the problem.

As we lower the total effective workload in the grid system, the second effect i.e. decreasing the number of available providers due to the mismatch in the reliability criterion, has less impact. Thus the optimum failure rate is lower at lower workloads and occurs at higher γ-values. This is indeed what we observe, for both churn distributions.

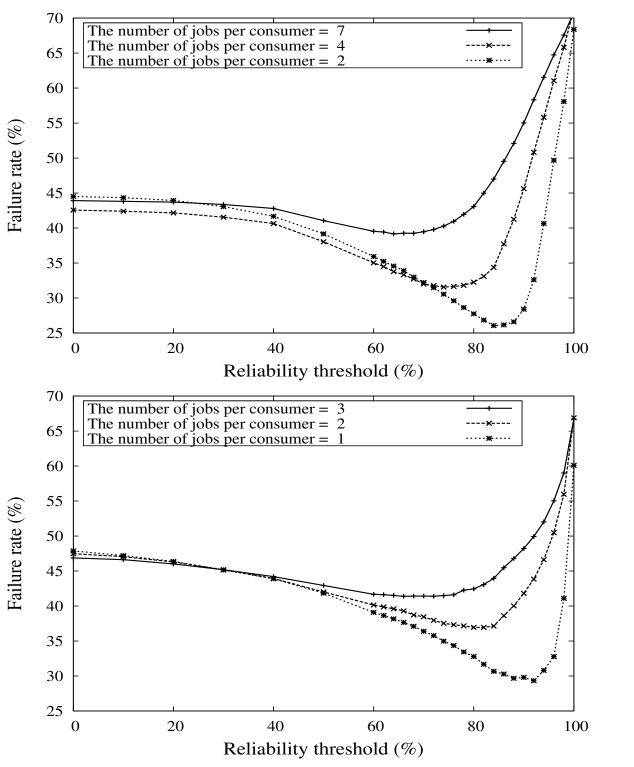

Fig. 6. Scenario III: The effect of workload on the failure rate for different (γ). The `microsoft99` (top panel) and `p105` (bottom panel) churn distributions and 5 provider groups.

V. Future work

Future work in this area of study includes introduction more refined definitions of reliability, more refined criteria for screening the providers that also take into account the lapsed time in the uptime interval. In addition the formulation of quality of service guarantees needs to be investigated.

VI. Conclusion

In desktop and peer-to-peer dynamic grids job failure due to churn in the grid system are inevitable. There is a need to minimize the impact of these job failures on the quality of service provided by such grids. In order to find effective approaches that can alleviate the impact of churn we need to model churn. In our work we have used models that are available in the literature.

In the context of the Grid Economics simulator framework we have programmed these churn models and developed resource allocation scheme based on first-price-sealed-bid auctions. We have then used this to investigate an algorithm, which we refer to as the threshold algorithm, to alleviate the impact of churn. The algorithm is based on a simple criterion which limits the bids only to reliable providers, where reliability is defined as having a reliability index above a certain threshold. The reliability index is a simple measure of availability based on average aggregate uptime.

We analyse experiments in a number of scenarios and arrive at the conclusions that firstly the effect of the threshold algorithm is fairly transparent and secondly it has the potential of improving failure rates significantly if the effective workload in the grid system is not too high.

References

[1] D. P. Anderson, "A system for public-resource computing and storage," in *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004, pp. 4–10. [Online]. Available: http://boinc.berkeley.edu/grid-paper-04.pdf

[2] ——, "BOINC: A system for public-resource computing and storage," in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*. IEEE Computer Society, 2004, pp. 4–10.

[3] P. Kacsuk and G. Terstyanszky, "Special section: Grid-enabling legacy applications and supporting end users," *Future Generation Comp. Syst.*, vol. 24, no. 7, pp. 709–710, 2008.

[4] P. Kacsuk, A. C. Marosi, J. Kovacs, Z. Balaton, G. Gombs, G. Vida, and A. Kornafeld, "Sztaki desktop grid: A hierarchical desktop grid system," in *Proceedings of the Cracow Grid Workshop 2006*, Cracow (Poland), 2007.

[5] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: An experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.

[6] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2006, pp. 189–202.

[7] ——, "Characterizing churn in peer-to-peer networks," Univ. of Oregon, Tech. Rep. CIS-TR-2005-03, May 2005.

[8] S. Y. Ko, I. Hoque, and I. Gupta, "Using tractable and realistic churn models to analyze quiescence behavior of distributed protocols," in *SRDS '08: Proceedings of the 2008 Symposium on Reliable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 259–268.

[9] D. Nurmi, J. Brevik, and R. Wolski, "Modeling machine availability in enterprise and wide-area distributed computing environments," in *In Euro-Par05*, 2003, pp. 432–441.

[10] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the condor experience," *Concurrency and Computation: Practice and Experience*, vol. 17, pp. 323 – 356, 2005.

[11] Hawkeye, "Monitoring and management tool for distributed systems," 2006, associated with the Condor Project. [Online]. Available: http://www.cs.wisc.edu/condor/hawkeye/

[12] D. Kondo, A. Andrzejak, and D. P. Anderson, "On correlated availability in internet-distributed systems," in *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, ser. GRID '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 276–283. [Online]. Available: http://dx.doi.org/10.1109/GRID.2008.4662809

[13] C. Elkan, "Using the triangle inequality to accelerate k-means," in *ICML*, 2003, pp. 147–153.

[14] A. Andrzejak, D. Kondo, and D. P. Anderson, "Ensuring collective availability in volatile resource pools via forecasting," in *19th IEEE/IFIP Distributed Systems: Operations and Management (DSOM 2008)*, Samos Island, Greece, September 2008, pp. 149–161. [Online]. Available: http://www.zib.de/andrzejak/

[15] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 398–407. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2010.71

[16] D. Kondo, M. Taufer, C. L. B. III, H. Casanova, and A. A. Chien, "Characterizing and evaluating desktop grids: An empirical study," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS?04)*. IEEE Computer Society, 2004, pp. 1 – 10.

[17] B. Ranjita, S. Stefan, and V. Geoffrey, "Understanding availability," 2003, pp. 256–267. [Online]. Available: http://www.springerlink.com/content/ehcfgw36n3j1ypr6

[18] R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson, and D. A. Patterson, "The interaction of parallel and sequential workloads on a network of workstations," *SIGMETRICS Perform. Eval. Rev.*, vol. 23, no. 1, pp. 267–278, 1995.

[19] W. Rich, S. Neil, and H. Jim, "Predicting the cpu availability of time-shared unix systems on the computational grid," in *HPDC '99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 1999, p. 12.

[20] K. Abdelkader and J. Broeckhove, "Pricing computational resources in a dynamic grid," *International Journal of Grid and Utility Computing (IJGUC)*, vol. 1, pp. 205 – 215, 2009.