A model of pilot-job resource provisioning on production grids

Tristan Glatard and Sorina Camarasu-Pop

CREATIS - CNRS UMR 5220 - INSERM U1044 - Université Lyon 1 - INSA Lyon
69621 Villeurbanne, FRANCE

Abstract

Pilot-job systems emerged as a computation paradigm to cope with the heterogeneity of large-scale production grids, greatly reducing fault ratios and middleware overheads. They are now widely adopted to sustain the computation of scientific applications on such platforms. However, a model of pilot-job systems is still lacking, making it difficult to build realistic experimental setups for their study (e.g. simulators or controlled platforms). The variability of production conditions, background loads and resource characteristics further complicate this issue. This paper presents a model of pilot-job resource provisioning. Based on a probabilistic modeling of pilot submission and registration, the number of pilots registered to the application host and the makespan of a divisible-load application are derived. The model takes into account job failures and it does not make any assumption on the characteristics of the computing resources, on the scheduling algorithm or on the background load. Only a minimally invasive monitoring of the grid is required. The model is evaluated in production conditions, using logs acquired on a pilot-job server deployed in the biomed virtual organization of the European Grid Infrastructure. Experimental results show that the model is able to describe accurately the number of registered pilots along time periods ranging from a few hours to a few days and in different pilot submission conditions.

Keywords: Production grids, pilot-job systems, probabilistic modeling.

1. Introduction

Large-scale production grids such as the European Grid Infrastructure (EGI[1]) are now used by a variety of applications benefiting from computing power and storage space provided by federations of computing centers. Shared by thousands of users, these infrastructures have become complex systems, providing heterogeneity, high variability, low reliability and high latencies as a downside of the huge amount of provided resources. On EGI, dozen of minutes of latency with similar standard-deviations and fault ratios of 20% are a common toll to access some of the 160,000 provided CPUs.

Models are lacking to describe application behavior on these systems and building them is difficult for the following reasons. Firstly, because of the wide scale and dynamicity of the system (downtime periods, new sites joining the infrastructure), it is difficult to get precise information about the platform characteristics, which is required by many simulation or analytical models. Secondly, production grids are shared among many users and load patterns of such systems are still being investigated [1]. In addition, system administration is distributed among several sites and it is not always clear which algorithms are used and how they are parametrized.

Consequently, research works targeting such systems are bound to conduct cumbersome experiments on the production infrastructure, with questionable reproducibility. In particular, experimental results very much depend on the availability of resources at the time of the experiments and on the activity of other users (a.k.a. background load). As a side-effect, application performance is hardly predictable and it remains challenging to forecast potential runtime issues or bottlenecks.

Among application-level computing frameworks, pilot jobs provide a submission scheme where tasks are no longer pushed through the grid scheduler but are put in a master pool and pulled by pilots running on computing nodes. The architecture of a pilot-job framework is described on Figure 1. Instead of being directly submitted to the grid middleware, application tasks are sent to a pool located on the application host and set up by the pilot master. Based on information reported by the master, a pilot controller submits pilots (also called agents, workers or generic jobs) to provision computing resources. If a pilot happens to reach a computing resource, it registers back to the pilot master and becomes available for the application. Then the mas-

[1] `http://www.egi.eu/`

ter can distribute tasks to the available pilots according to some scheduling policy.

In most clusters the firewall configuration forbids incoming connections to the computing resources. To overcome this issue pilots usually send periodical heartbeats to the master, requesting for tasks. User accounting and the handling of file permissions require that application tasks are executed with the identity of the user that started the application. However, pilots are often submitted by a different identity (e.g., a robot) provisioning resources for a group of users. In this case the identity should be switched before executing the tasks, for instance using a tool like gLExec[2].

As evoked in [2] two main problems have to be addressed by pilot-job systems. On the one hand, resource provisioning (gray plain arrows on Figure 1) aims at controlling pilot submission so that application requirements are fulfilled and computing resources are not kept idle. On the other hand, scheduling (white dashed arrow on Figure 1) aims at selecting appropriate pilots among registered ones to execute application tasks. With no suitable model available, solutions to these issues can hardly be investigated.

Although pilots are still submitted through the regular grid middleware, such a pull model greatly improves fault-tolerance. Indeed failures have a limited impact because failed tasks can directly be executed by other pilots (faulty pilots are removed). The overhead is also reduced since tasks are directly scheduled by the master on the pilots without going through the whole grid middleware again. In fact, it is hardly feasible to conduct successfully significant experiments on infrastructures such as EGI without using such a system. Most application groups, including High-Energy Physics experiments now rely on them.

In this paper, we target the modeling of pilot-job resource provisioning. Assuming that the pilot submission mechanism is known, our goal is to model pilot registration. Following previous works, we adopt a probabilistic approach relying on the latency distribution to derive statistics about the modeled quantities. Related work is reported in Section 2, the model is detailed in Section 3 and it is evaluated on the production infrastructure in Section 4.

[2] `https://twiki.cern.ch/twiki/bin/view/EGEE/`

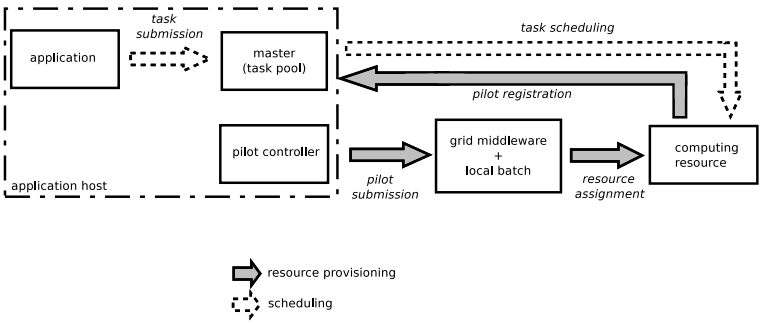


Figure 1: Pilot-job execution framework: resource provisioning and task scheduling.

2. Related work

Several pilot-job frameworks have been proposed in the last few years. In particular, systems interfaced with EGI include DIANE [3], WISDOM-II [4, 5], ToPoS[3], BOINC [6], PANDA [7] and DIRAC [8]. Condor[4] also has its pilot-job submission framework called glideIn [9].

There has been abundant literature on task scheduling for master/slave applications, including works on heterogeneous platforms. In particular, complexity results and algorithms are presented in [10] and a model of master/slave application performance is described in [11]. Those works only target scheduling and do not consider resource provisioning which is our main concern here.

Singh *et al.* propose in [12] a model used to minimize both the scheduling and the allocation (provisioning) cost. The provisioning cost is defined from an overhead cost and a resource cost per unit of time. Although the model shows the interest of using resource provisioning instead of the best effort, assumptions are slightly different from ours since the described model assumes that the availability of processors on a given resource is known. Instead we focus on determining resource availability, i.e., pilot registration time.

More closely related to our problem is the work described in [13] which models pilot-job applications (denoted global computing) to simulate the DIRAC system. The results are quite accurate (a 12% error on the makespan prediction is reported) but the simulation considers a static reliable platform,

[3] `https://wiki.nbic.nl/index.php/ToPoS`
[4] `http://www.cs.wisc.edu/condor/`

with a defined set of clusters and worker nodes per cluster. Under our assumptions, it is difficult to have an idea on those numbers and failures are very likely to occur. Introducing failure probabilities in the simulation such as done in [14] could overcome some limitations. However, to date, no realistic simulator model for very large scale platforms such as EGI has been validated. Instead, we currently aim at deriving macroscopic models describing the global behavior of the infrastructure with little assumption on its underlying structure.

To the best of our knowledge, no analytical or simulation model has been proposed to describe pilot-job applications on large-scale heterogeneous dynamic platforms such as EGI. The following Section attempts to do that.

3. Modeling

The model proposed here intends to describe the behavior of resource provisioning at a global level, i.e., without any detailed information on the underlying platform. In particular, no assumption is made on the number or performance of computing resources, on the job scheduling algorithm or on the nature of the background load. Instead, the grid is seen as a black box delaying job execution with a duration called *latency*. The latency is defined as the total duration between the submission of a job and the beginning of its execution on a computing resource. It is described by a probabilistic distribution which is often not stationary. Other problems relying on the same black-box approach are found in [15] and latency observations are reported in [16].

It will be assumed that jobs face independent and identically distributed (*iid*) latencies. This implies that any phenomenon resulting from interactions among jobs submitted by the modeled process is ignored, for instance network contention collapse, saturation or thrashing of grid middleware services. This is realistic as long as the modeled process is of negligible size with respect to the computing infrastructure. Given the size of the considered infrastructure (EGI) this assumption reasonably holds, except on the application host from which jobs are submitted. To cope with this issue we show in Section 3.1 how the modeling can be adjusted to the *iid* assumption in case of sequential pilot submissions.

Given a set of submitted pilots, the model then aims at estimating (i) the evolution along time of the number of registered pilots and (ii) the makespan of the application, i.e., the duration between the submission of the first task

and the completion of the last task. These are respectively described in Sections 3.2 and 3.3.

3.1. Job latency for sequential job submission

In some cases pilot jobs are sequentially submitted to avoid overloading the schedulers. This is for instance the case in the setup considered in Section 4. In this case the *iid* assumption does not hold but it is possible to resume to it as shown below. In the following, capitals are random variables and lowercases are fixed values. L denotes the latency and it is split into a submission term S and a scheduling/queuing term G ($L = S + G$). The probabilistic density function (*pdf*) of a random variable X is written f_X and the cumulative (*cdf*) is written F_X. The convolution operator is denoted $*$ and f_X^{*k} is f_X convolved k times with itself[5].

In the sequential submission of n jobs, the submission of job $i \in [\![1, n]\!]$ is delayed by the sum of the submission times of its predecessors. Therefore, the submission term of L_i, the latency of job i, depends on i:

$$L_i = S_i + G.$$

If we consider that $S_i = iS$ then $f_{Si} = f_S^{*i}$ and $f_{Li} = f_S^{*i} * f_G$. Thus L_i cannot be assumed *iid*. Instead, we consider that the job rank i is randomly uniformly distributed between 1 and n, so that f_{Si} does not depend on i:

$$\begin{aligned} \forall i \in [\![1, n]\!], \quad f_{Si}(t) &= \frac{1}{n}\left(\sum_{k=1}^{n} f_{kS}(t)\right) \\ &= \frac{1}{n}\left(\sum_{k=1}^{n} f_S^{*k}(t)\right) = f_{\tilde{S}}(t). \end{aligned}$$

This requires that S is stationary during the submission phase, which assumes that no significant burst period occurs during job submission. Handling bursts is a problem currently under study and exploiting works such as [17] may further improve our modeling.

Under these assumptions the *pdf* of L_i is also independent from i:

$$\begin{aligned} f_{Li}(t) = f_{\tilde{L}}(t) &= f_{\tilde{S}}(t) * f_G(t) \\ &= \frac{1}{n}\left(\sum_{k=1}^{n} f_S^{*k}(t)\right) * f_G(t), \end{aligned}$$

[5] $f_X(t) = P(X = t)$, $F_X(t) = P(X < t)$ and $f_{X+Y} = f_X * f_Y$

and its *cdf* is:

$$F_{\tilde{L}}(t) = \int_0^{\infty} \frac{1}{n} \left(\sum_{k=1}^{n} f_S^{*k}(u) \right) F_G(t-u)du. \quad (1)$$

For consistency purposes $F_{\tilde{L}}$ is used in the following Sections. If the model is applied to a system where jobs are independently submitted then F_L can be directly used instead.

3.2. Number of registered pilots

In this section $t = 0$ denotes the submission time of the first pilot. The goal is to determine N, the number of registered pilots at a given instant. We assume that pilot jobs may fail and never register. For instance, it happens in case of network firewall issues or other configuration problems in the grid middleware. The pilot error ratio is denoted ρ and failed jobs are assumed to have an infinite latency. Thus the probability that a pilot submitted at $t = 0$ registers before time t is $\tilde{F}_{\tilde{L}}(t) = (1-\rho)F_{\tilde{L}}(t)$[6].

Although this simple fault model probably does not hold on platforms such as the ones mentioned in [18, 19], it is here supported by evidence found in [16] (Fig. 9), based on an analysis of the trace of 33 millions of jobs submitted between 2005 and 2007 on the EGI. Results show that using such a $(1-\rho)F_{\tilde{L}}$ fault model instead of the actual fault distribution only has a small impact on the considered parameter estimation.

If n pilots are submitted in a single submission wave starting at $t = 0$, then the probability to have k pilots registered at time t is:

$$f_N(k,t) = \binom{n}{k} \tilde{F}_{\tilde{L}}(t)^k (1-\tilde{F}_{\tilde{L}}(t))^{(n-k)},$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. Indeed, any k pilots among n may be registered at time t (i.e., $\tilde{L} \leq t$ for k pilots, which occurs with probability $\tilde{F}_{\tilde{L}}(t)^k$), while the other $(n-k)$ are still being scheduled or queued ($\tilde{L} \geq t$ for $n-k$ pilots, probability $(1-\tilde{F}_{\tilde{L}}(t))^{n-k}$). We can notice that at a given instant N follows a binomial distribution of parameter $\tilde{F}_{\tilde{L}}(t)$. Thus its expectation and variance are:

$$E_N(t) = n\tilde{F}_{\tilde{L}}(t) \quad (2)$$

$$\sigma_N(t)^2 = n(1-\tilde{F}_{\tilde{L}}(t))\tilde{F}_{\tilde{L}}(t). \quad (3)$$

[6]Note that $\tilde{F}_{\tilde{L}}$ is not a *cdf* since $\lim_{t\to+\infty} \tilde{F}_{\tilde{L}}(t) \neq 1$.

From Equation (2) we have $\lim_{t\to\infty} E_N(t) = n(1-\rho)$, i.e., after enough time, all the submitted pilots will register, up to the failure ratio. Moreover, $\lim_{t\to\infty} \sigma_N^2(t) = n\rho(1-\rho)$. Note that this limit increases from $\rho = 0$ to $\rho = 1/2$ and decreases from $\rho = 1/2$ to $\rho = 1$: the worst model accuracy is obtained for $\rho = 1/2$.

In practice, pilots are often submitted in several successive waves. In this case, the total number of registered pilots is $N = N_1 + N_2 + \ldots + N_p$, where p is the number of submission waves and N_i is the number of registered pilots from wave i. Submission waves start at times $t_1, \ldots, t_p$ and submit $n_1, \ldots, n_p$ pilots. Since the latency cannot be assumed stationary across waves, the latency at wave i is written $\tilde{L}_i$, ρ_i is the error ratio of the pilots submitted in wave i and $\forall t \geq t_i$, $\tilde{F}_{\tilde{L}i}(t) = (1-\rho_i)F_{\tilde{L}i}(t)$. We also set $\forall t < t_i$, $\tilde{F}_{\tilde{L}i}(t) = 0$.

We have:

$$f_N(k,t) \quad = \quad f_{N1}(k,t) * f_{N2}(k,t) * \ldots * f_{Np}(k,t),$$

and by linearity from Equations (2) and (3):

$$E_N(t) \quad = \quad \sum_{i=1}^{p} n_i \tilde{F}_{\tilde{L}i}(t) \tag{4}$$

and

$$\sigma_N(t)^2 \quad = \quad \sum_{i=1}^{p} n_i (1 - \tilde{F}_{\tilde{L}i}(t)) \tilde{F}_{\tilde{L}i}(t). \tag{5}$$

The asymptotic behavior is the following:

$$\lim_{t\to+\infty} E_N(t) = \sum_{i=1}^{p} n_i(1-\rho_i) \tag{6}$$

$$\text{and } \lim_{t\to+\infty} \sigma_N^2(t) = \sum_{i=1}^{p} n_i \rho_i (1-\rho_i).$$

3.3. Makespan modeling

The application is defined by its total amount of work time w_0 (CPU time and data transfers) on the average grid resource. It is assumed to be a divisible load [20], i.e., the worktime can be split into any number of independent tasks. The pilots are assumed generic, i.e., they can execute any task.

Linear speed-up w.r.t. the number of available pilots is also assumed. This considers that an application running on a given set of pilots would behave as if all the pilots were achieving the average performance, which is asymptotically realistic.

Let $W(t)$ denote the remaining work time at time t ($W(0) = w_0$). According to the linear speed-up assumption, we have:

$$dW = -N(t)dt,$$

so that the remaining amount of work at a given instant is:

$$W(t) = \max(w_0 - \int_0^t N(u)du, 0),$$

and given Equation (4):

$$E_W(t) = \max(w_0 - \sum_{i=1}^{p} n_i \int_0^t \tilde{F}_{\tilde{L}i}(u)du, 0).$$

The expectation E_M of the makespan M of an application is then obtained by solving the following in t:

$$w_0 - \sum_{i=1}^{p} n_i \int_0^t \tilde{F}_{\tilde{L}i}(u)du = 0.$$

4. Experiment and results

The goal of this experiment is to evaluate in production conditions the validity of the pilot registration model, i.e., Equations 4 and 5. The makespan model was already evaluated in [21].

4.1. Experimental data

For this experiment, data was collected from the production service operated at the Creatis CNRS laboratory[7]. This service supports the execution of about 10 medical imaging applications run by approximately 15 users on the biomed virtual organization (VO) of the European Grid Infrastructure. This

[7] `http://www.creatis.insa-lyon.fr`

VO has access to some 250 cluster batch queues managing a total of 92,000 CPUs. Every user has his/her own dedicated DIANE pilot master, shared by all his/her runs. Pilots are submitted by waves following algorithm 1. The number of waves (i.e., iterations of the while loop where $n \neq 0$) and the number of pilots submitted in each wave depend on the number of tasks submitted by the user.

Algorithm 1 Algorithm controlling the submission of DIANE pilots

```
init=10, defaultSleep=400, s=defaultSleep, maxSub=300, factor=5
start DIANE master
submit init pilots
while master is alive do
  sleep s seconds
  n = number of tasks waiting in master
  submit sub=min(maxSub,n) pilots
  s = defaultSleep+sub*factor
end while
```

As shown on Table 1, six logs belonging to different users are available. When these logs were acquired users were using different kinds of applications (Monte-Carlo and parameter sweep) and were having different activities (debugging VS production). Logs have different numbers of submission waves, submitted/registered pilots and were collected on different time frames. Logs were post-processed to remove time intervals where many pilots were submitted but none registered, indicating a critical problem on the platform (e.g. user credentials expired).

The grid latency (F_{Li} and ρ_i) was measured from the round-trip times of probe jobs. Ten (10) "hostname" probes were permanently maintained inside the system. Each time a probe completed a new one was submitted. These probes were submitted using similar parameters as used on the production platform, yet from a different machine. Given the size and throughput of the EGI this monitoring is considered minimally invasive.

For each submission wave i, the grid latency was estimated from the probes submitted up to 1 hour before the submission of the first pilot of wave i. This was empirically tuned to consider a sufficient amount of probes.

For each log the total number of probes used to compute the entire model (including all waves) and the overall error rate (total number of failed probes in all waves divided by the total number of submitted probes in all waves) is

reported in Table 2. Logs are ordered by increasing values of the error rate. From these probes the computation of E_N and σ_N was done straight from Equations (4) and (5).

4.2. Results and discussion

Figure 2, 3 and 4 plot the evolution of the model (E_N and σ_N) along time in thin dashed lines and the measured data in plain bold lines. The measured number of registered pilots is written n_R, and the model error is written ϵ. The model error was computed as $\epsilon(t) = |E_N(t) - n_R(t)|$. For each log E_N is plotted with the measured number of registered pilots on the left graph and σ_N is plotted with the model error on the right graph. The number of submitted pilots is also shown.

As shown on the left parts of Figures 2, 3 and 4, E_N and n_R consistently have very similar shapes, indicating a strong correlation between E_N and the measured number of registered pilots. The step shape coming from wave submissions is properly captured by the model. Even when the probe error rate increases (Figure 4(a) and 4(b)) the model correctly follows the variations of the input data.

The first column of Table 3 shows the average model error $\bar{\epsilon}$ and the second column compares it to the average number of registered pilots $\bar{n}_R$. In most cases the relative error is below 10%, which on a production system indicates very good model performance. The third log has a larger 16% relative error which is explained by wrong estimations of the error rate. Indeed as the left part of Figure 3(a) shows, the error is very important at the wave asymptote, which indicates a poor estimation of ρ_i values (see Equation (6)). This is a

Table 1: Experimental data.

Log name	Number of submission waves (p)	Total numbers of submitted pilots ($\sum_{i=1}^{p} n_i$)	Total number of registered pilots	Log acquisition duration
backup-a23055	11	489	324	3.8 hours
backup-sarra	78	716	574	12 hours
backup-rafael	29	3220	2622	2.8 days
backup-daniel	79	5202	3936	7.4 days
backup-ting	50	3554	2965	2.9 days
backup-sorina	31	163	127	9.4 hours

Table 2: Probes used to build the model.

Log name	Number of probes used in model computation	Probe error rate
backup-a23055	127	0.0078
backup-sarra	390	0.046
backup-rafael	424	0.078
backup-daniel	863	0.097
backup-ting	391	0.12
backup-sorina	244	0.16

Table 3: Model accuracy.

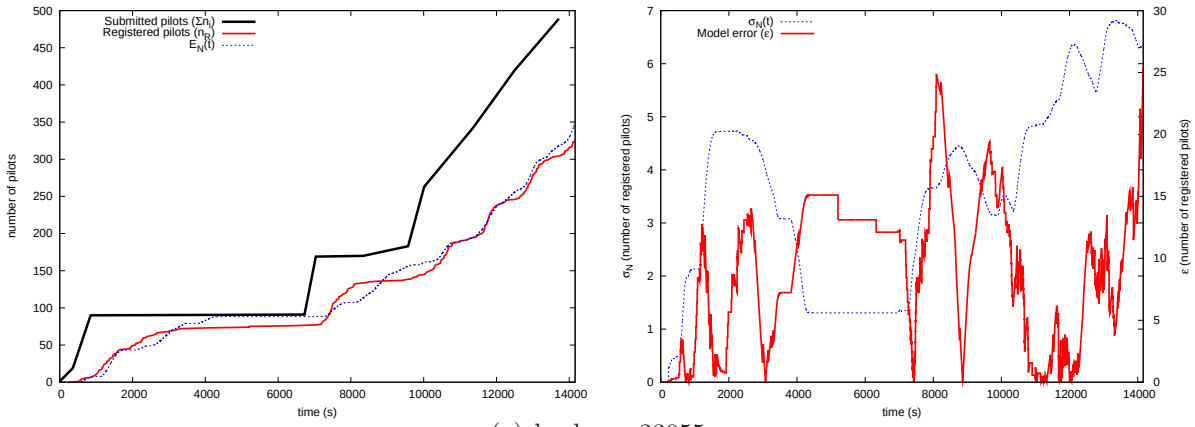
Log name	$\bar{\epsilon}$	$\frac{\bar{\epsilon}}{\bar{n}_R}$	$\frac{\bar{\epsilon}}{\bar{\sigma}_N}$
backup-a23055	9.2	0.07	2.6
backup-sarra	37.6	0.09	8.4
backup-rafael	163.7	0.16	21.6
backup-daniel	144.3	0.04	6.6
backup-ting	58.2	0.03	3.9
backup-sorina	7.8	0.08	1.7

typical problem in the estimation of rare events and should be investigated with dedicated statistical tools.

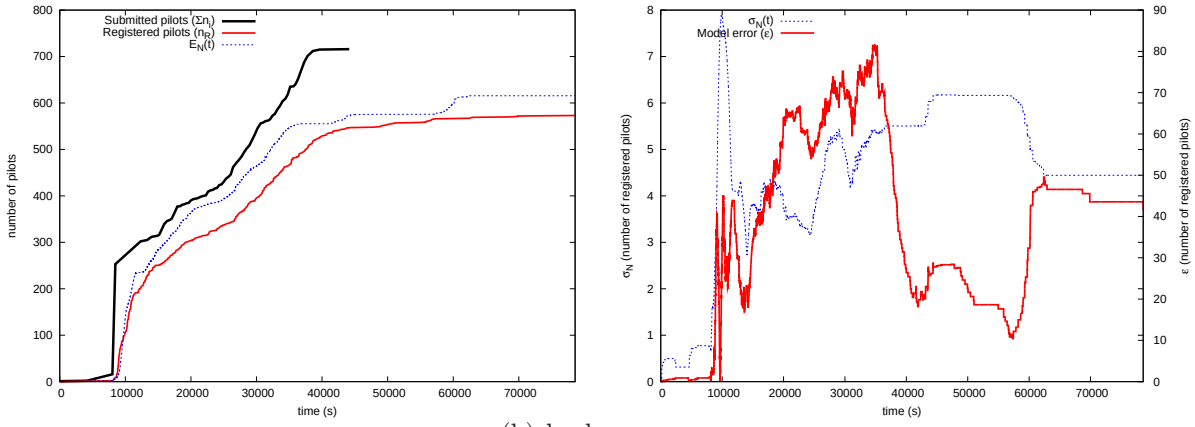
Although the right parts of Figures 3(a) and 3(b) show some correlation between σ_N and ϵ, this is not obvious in the other plots (note that the two curves are plotted on different y scales). Yet the third column of Table 3 shows that the ratio between $\bar{\epsilon}$ and the average σ_N value $\bar{\sigma}_N$ stays under 8.4 for 5 of the logs, which is an acceptable bound. The 21.6 ratio observed for log `backup-rafael` originates in the already-mentioned error rate estimation issue. Overall σ_N provides some indication on the accuracy of the model although it cannot be used as a quantitative reference as E_N.

5. Conclusion

The model is able to describe consistently the registration of a set of submitted pilot jobs to the application host. It makes no assumption on the infrastructure characteristics (number of computing resources, network characteristics, background load, scheduling algorithm) and the only required

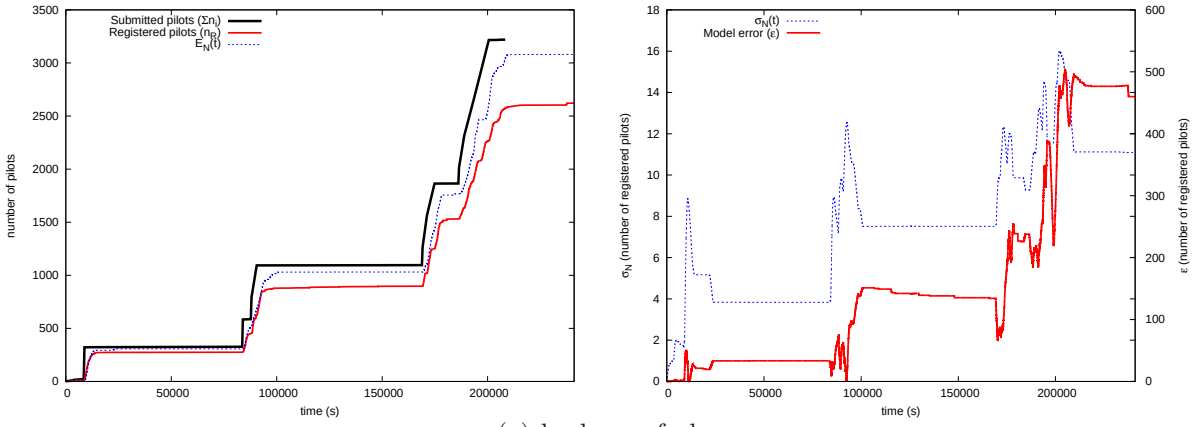


(a) backup-a23055

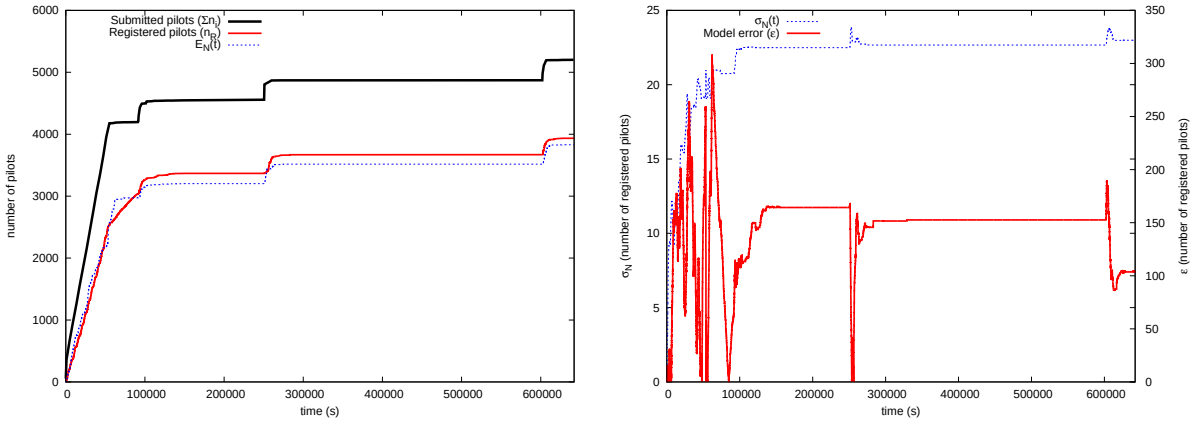


(b) backup-sarra

Figure 2: Model VS experimental data for logs 1 and 2.

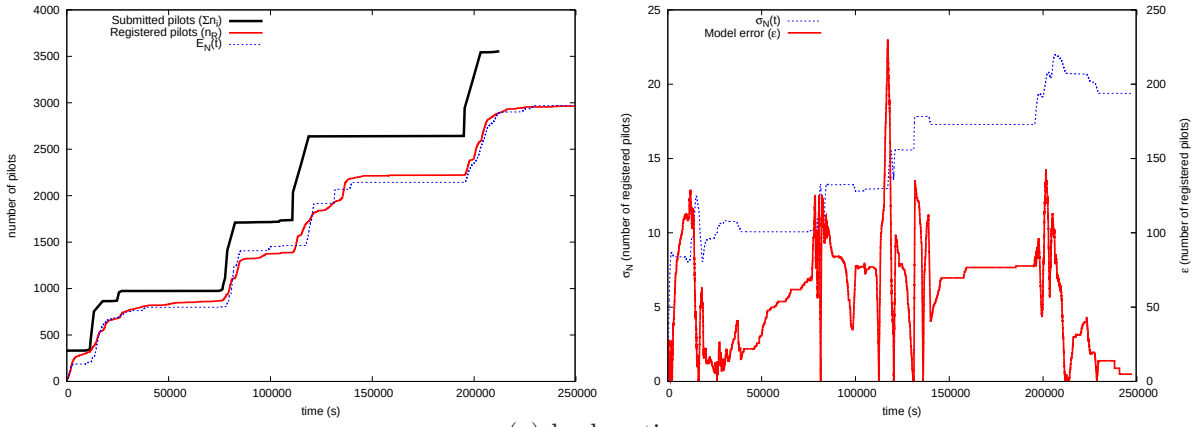


(a) backup-rafael

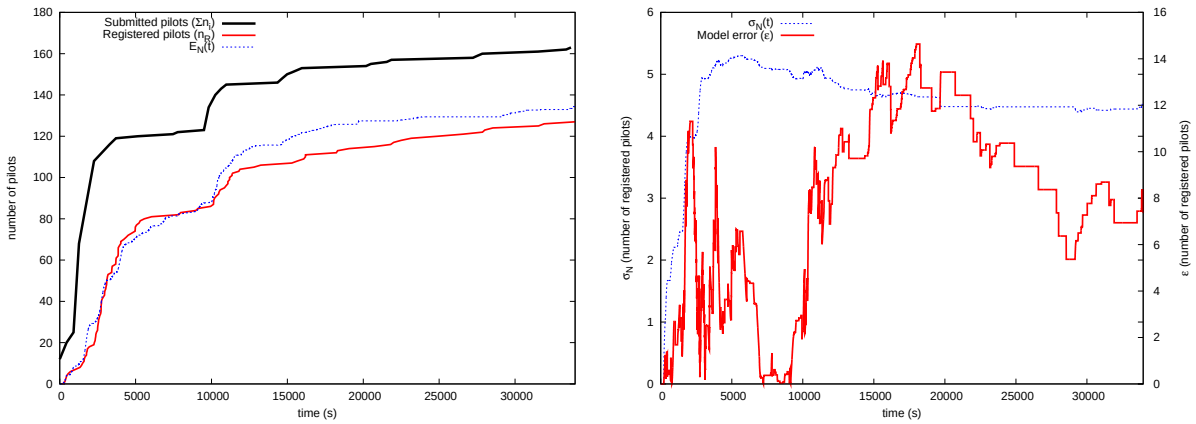


(b) backup-daniel

Figure 3: Model VS experimental data for logs 3 and 4.



(a) backup-ting



(b) backup-sorina

Figure 4: Model VS experimental data for logs 5 and 6.

parameter (an estimation of the latency) is captured with a minimally invasive monitoring system.

The main limitation of the approach is its dependence to the job latency estimation. Although a new latency estimate is considered for each pilot submission wave, rapid changes of the grid conditions during the submission of a pilot wave would hamper the accuracy of the model. Moreover, inaccuracies in the estimation of job errors significantly disturb the model.

The practical implications of this model are important since it is able to deal with the dynamicity and the background load of a production grid, which are the main causes hampering the reproducibility of experiments on production platforms. With little parametrization (job latency estimates), this model can be used to reproduce in controlled conditions the availability of resources on a production platform. For instance, it could be used to build a realistic dynamic platform model in simulators such as SimGrid [22] or on experimental platforms.

From a conceptual point of view, the modeling initiated here could be extended to describe the interaction of load balancing strategies with pilot submission strategies. Better understanding these interactions could help designing load balancing methods that can cope with the dynamicity of production platforms, which is still hampering current approaches. In particular, testing the influence of pilot submission algorithms on the dynamic partitioning algorithm proposed in [23] for Monte-Carlo simulations is part of our future work.

6. Acknowledgement

We warmly thank Jakub Mościcki for his assistance with the DIANE framework and Thomas Grenier for fruitful discussions. We are also grateful to the European Grid Infrastructure for providing the infrastructure and related user support. This work is partially funded by France-Grilles[8].

[1] X. Zhang, C. Furtlehner, J. Perez, C. Germain, M. Sebag, Toward Autonomic Grids: Analyzing the Job Flow with Affinity Streaming, in: Proceedings of 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD'2009), 2009.

[8] http://www.france-grilles.fr/

[2] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde, Falkon: a Fast and Light-weight tasK executiON framework, in: Proceedings of the 2007 ACM/IEEE conference on Supercomputing (SC'07), 2007, pp. 1–12.

[3] J. T. Mosciki, Distributed analysis environment for HEP and interdisciplinary applications, Nuclear Instruments and Methods in Physics Research A 502 (2003) 426429.

[4] S. Ahn, K. Namgyu, L. Seehoon, H. Soonwook, N. Dukyun, B. Koblitz, V. Breton, H. Sangyong, Improvement of Task Retrieval Performance Using AMGA in a Large-Scale Virtual Screening, in: NCM'08, 2008, pp. 456–463.

[5] N. Jacq, J. Salzeman, F. Jacq, Y. Legre, E. Medernach, J. Montagnat, J. Maass, M. Reichstadt, H. Schwichtenberg, M. Sridhar, V. Kasam, M. Zimmermann, M. Hofmann, V. Breton, Grid-enabled Virtual Screening against malaria, Journal of Grid Computing 6 (1) (2008) 29–43.

[6] P. Kacsuk, Z. Farkas, G. Fedak, Towards making BOINC and EGEE interoperable, in: 4th International Conference on e-Science, Indianapolis, USA, 2008.

[7] T. Maeno, PanDA: distributed production and distributed analysis system for ATLAS, Journal of Parallel Computing 119 (6) (2008) 062036 (4pp).

[8] V. Garonne, A. Tsaregorodtsev, I. Stokes-Rees, DIRAC: A scalable lightweight architecture for high throughput computing, in: 5th International Workshop on Grid Computing (GRID'04), IEEE Computer Society, 2004, pp. 19–25.

[9] I. Sfiligoi, glideInWMS: a generic pilot-based workload management system, Journal of Physics: Conference Series 119 (6).

[10] O. Beaumont, A. Legrand, Y. Robert, The Master-Slave Paradigm with Heterogeneous Processors, IEEE Transactions on Parallel and Distributed Systems (TPDS) 14 (9) (2003) 897–908.

[11] G. Shao, F. Berman, R. Wolski, Master/Slave Computing on the Grid, in: 9th Heterogeneous Computing Workshop (HCW), 2000, p. 3.

[12] G. Singh, C. Kesselman, E. Deelman, A provisioning model and its comparison with best-effort for performance-cost optimization in grids, in: Proceedings of the 16th international symposium on High performance distributed computing (HPDC'07), 2007, pp. 117–126.

[13] E. Caron, V. Garonne, A. Tsaregorodtsev, Definition, modelling and simulation of a grid computing scheduling system for high throughput computing, Future Generation Computer Systems (FGCS) (23) (2007) 968–976.

[14] V. Berten, E. Jeannot, Modeling Resubmission in Unreliable Grids: the Bottom-Up Approach, in: Seventh International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (Heteropar'09), 2009.

[15] T. Glatard, Description, deployment and optimization of medical image analysis workflows on production grids, Ph.D. thesis, Université de Nice Sophia-Antipolis, Sophia Antipolis, France (Nov. 2007).

[16] D. Lingrand, J. Montagnat, J. Martyniak, D. Colling, Analyzing the EGEE production grid workload: application to jobs submission optimization, in: 14th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'09), Springer, 2009, pp. 37–58.

[17] T. Ngoc Minh, L. Wolters, Modeling Job Arrival Process with Long Range Dependence and Burstiness Characteristics, in: CCGrid'09, 2009, pp. 324–330.

[18] B. Schroeder, G. A. Gibson, A large-scale study of failures in high-performance computing systems, DSN (2006) 249–258.

[19] S. Fu, C.-Z. Xu, Exploring event correlation for failure prediction in coalition of clusters, in: Supercomputing, 2007.

[20] V. Bharadwaj, D. Ghose, T. G. Robertazzi, Divisible load theory: A new paradigm for load scheduling in distributed systems, Cluster Computing 6 (2003) 7–17.

[21] T. Glatard, S. Camarasu-Pop, Modelling pilot-job applications on production grids, in: 7th international workshop on Algorithms, Models

and Tools for Parallel Computing on Heterogeneous Platforms (Heteropar 09), Delft, The Netherlands, 2009.

[22] H. Casanova, A. Legrand, M. Quinson, Simgrid: A generic framework for large-scale distributed experiments, International Conference on Computer Modeling and Simulation (2008) 126–131.

[23] S. Camarasu-Pop, T. Glatard, J. T. Moscicki, H. Benoit-Cattin, D. Sarrut, Dynamic partitioning of gate monte-carlo simulations on EGEE, Journal of Grid Computing 8 (2) (2010) 241–259.