Modeling user submission strategies on production grids
Diane Lingrand, Johan Montagnat, Tristan Glatard

To cite this version:
ABSTRACT

Production-grid users experience many system faults as well as high and variable latencies due to the scale, complexity and sharing of such infrastructures. To improve performance, they adopt different submission strategies, that are potentially aggressive for the infrastructure.

This work studies the impact of three different strategies. It is based on a probabilistic modeling of these strategies which are evaluated according to their performance, measured as the reduction of the latency expectation, and the infrastructure overhead, measured as the additional number of submitted jobs. A strategy cost criterion is then derived.

Experiments are performed using real workload traces collected from the EGEE production infrastructure. Under these conditions, a good balance between high performance and low overhead can be found.

Categories and Subject Descriptors
C.4 [PERFORMANCE OF SYSTEMS]: Modeling techniques

General Terms
Performance

Keywords
Grid Computing, Modelisation, Submission strategy

1. INTRODUCTION

Production grids are world-wide distributed systems, covering various network domains connected by WANs. They are highly variable environments federating sites with heterogeneous resources, configuration rules and access policies and servicing many users concurrently. Consequently, grids are characterized by high and non-stationary workloads. From a user point of view, the grid appears as a plethora of queues operated through different batch management systems, with different prioritization policies, that are difficult to exploit efficiently.

Unlike supercomputers no single grid scheduler can obtain a global view of the complete grid infrastructure. Instead, meta-schedulers coordinate the action of site schedulers, using heuristic that try to cope with (i) partial information and (ii) local independent scheduling policies that may interfere with the meta-scheduling objective. Moreover, job lifecycles include much more steps than scheduling and queuing (among others: credentials delegation, match-making, job submission language formats translation, logical-to-physical file names resolution, file replica selection, job monitoring system, grid information system). In the order of 10 machines, that all have to be reachable, may be involved in the job submission process.

The job submission system is subject to failures that can originate from network/connectivity problems, local configuration issues (authorization issues, differences in middleware versions, local environment, etc), workload variations (impacting all the components, not only the job queuing time), data access (data catalog querying, data transfer queuing, and the transfer itself), and scheduling issues. Nobody currently has a proven classification of all possible issues. Such a list is far from being trivial given the heterogeneity of the grid and the interoperability of complex stack of services.

As a consequence, grid jobs latency, measured as the duration between the beginning of a job submission and the time it starts executing, can be very high and prone to large variations. High fault ratios are often reported, as for example in [2]. High latency and faults impact the performance of applications, sometimes making grids unworthy for their tasks. Assuming that a single entity could achieve flawless scheduling on a world-wide scale production grid seems quite hazardous and we claim that a client-side error control is required as part of the scheduling, similarly to transport protocols implementing fault-tolerance mechanisms to deal with routing errors on the Internet.

As a matter of fact, it has been commonly observed on production systems that despite the middleware best efforts to address these problems internally, users do adopt different submission strategies controlled on the client side such as canceling jobs with too long latencies before resubmission or submitting several copies of the same job. These strategies are often empirical and not objectively evaluated. The motivation for this work is to provide an objective insight on these strategies. Strategies with a proven impact on applica-
tion performance and causing no critical overload of the job management system could then be integrated in the client side of the middleware to release the users of this burden.

The more variable the job latency, the more efficient multiple submission strategies. Although multiple submission strategies are effective from the user point of view, they are usually not appreciated from administrators since they are increasing the workload endured by the grid resource brokers and schedulers. Yet, little can be done to regulate their usage, especially on a wide scale distributed grid. Quantifying the impact of submission strategies on real-size grids is a difficult problem due to the complexity of such systems. In this paper, we adopt a probabilistic approach to model the grid responsiveness. Execution statistics are collected on the real-scale EGEE production grid over long periods of time to estimate probability laws of the job latencies. Several submission strategies are studied using these grid workload models: the single resubmission (section 4), the multiple submission (section 5) and a novel strategy that we call delayed resubmission (section 6).

These strategies are assessed considering two criteria: the performance improvement from users point of view (reduction in latency time) and the infrastructure load (number of tasks to manage). The delayed resubmission strategy demonstrates a very good balance between high performance and low load.

2. RELATED WORK

Many works study the scheduling of jobs on distributed systems at different levels. The algorithms are directly implemented in schedulers. For example, Subramani et al [16] compare two different scheduling schemes with respect to the slowdown computed as the ratio of (latency + runtime) over runtime. One is the “K-distributed scheme” which consists in submitting each job to the K least loaded sites. When the first job starts running, the other (K-1) jobs are canceled. The second one is the “K-Dual Model” which is an improvement of the K-distributed one, giving priority to locally submitted jobs, using two different queues. The performance analysis shows that the slowdown is better reduced by the K-Distributed than the K-Dual Queue. However, when considering lightly loaded sites and heavily loaded sites separately, there is an inversion with the K-Distributed scheme which does not occur with the K-Dual Queue one. Different values of K have been studied from 1 to 4, showing a decrease in the average slowdown. Authors also demonstrate that the impact of inaccuracies in user estimates of runtime is in favor of the proposed schemes and that they also perform better than the other schemes when considering the communication overheads.

Sabin et al [14] study the scheduling in a heterogeneous multi-sites environment. They evaluate different strategies including multiple submissions (k=4), conservative versus aggressive backfilling and the relative job efficacy for queuing priority. Different experimental results based on job traces showed that in the case of heterogeneous multi-sites (which is the case in a production grid), conservative backfilling is better than aggressive and that using the relative job efficacy for queuing priority improves performances.

Casanova [5] studies the multi-submission of jobs, considering that when the first job starts running, all other jobs waiting in queue are canceled. He observes that submitting all jobs several times increases their average performance; the users that are penalized are those that do not use multiple submission. The load on batch schedulers or network will not be critical if the number of multiple jobs is less than 30, assuming that submissions are uniformly distributed among sites and that the job inter-arrival is always 5 seconds. However, the author observes that the 2006 version of GRAM conducts to a bottleneck when using more than 3 multiple submissions during peak job submission.

In this paper we focus on large scale infrastructure characterized by highly variable latencies, such as the EGEE production grid\(^1\) on which experiments were conducted [7, 11]. Due to its unique scale, it enables challengingly large applications but it is known to be prone to faults and variable queuing times [9, 2, 1]. Yet, variability has also been acknowledged as a critical factor on other types of platforms, such as the knowARC grid\(^2\): for instance, the experiment reported in [13] shows variations up to 40 minutes in the grid time of an embarrassingly parallel applications and the work presented in [12] shows overheads ranging from 0 to 45 min. Earlier works conducted before the emergence of large-scale grids in the 2000s already pointed out the importance of variability, such as [13]: authors were then dealing with seconds-to-minutes variability values and the importance of this issue has dramatically increased in current grid infrastructures.

To address the complexity of modeling an heterogeneous grid infrastructure, we adopt a probabilistic approach. Statistical studies carried out in [4] and [6] and earlier works such as [10] or [9] are important contributions to workload modeling. However, those works mostly consider latencies from the perspective of a specific grid scheduler (if not a local site queue) and the actual match with the latency observed by an application still has to be validated. As explained above, several additional steps such as data transfers and proxy delegations are likely to disturb measures carried-out at the grid or local batch scheduler levels. Consequently, the monitoring approach adopted in this work relies on round-trip times of probes submitted to the grid from the user client, thus getting much closer to real application job submission conditions.

3. DEFINITIONS, ASSUMPTIONS AND REFERENCE DATA

We define the latency as the duration between the instant job submission instant and the beginning of its execution on a computation resource. It is modeled through a random variable \(R\). On the EGEE production grid, the latency distribution has been observed to be heavy-tailed, as reported, e.g., in [8]. We denote the fault (or outlier) ratio that commonly occur on a production grid as the real value \(\rho \in [0,1]\). Figure 1 plots \(F_R\), the cumulative density function (cdf) of the heavy-tailed random variable \(R\) and \(F_R(t) = (1 - \rho) F_R(t) + \rho\), the cumulative histogram of all latencies (normalized with respect to all submitted jobs, including outliers).

The probability for the latency of a job to be lower than a given threshold \(t\) depends on the probability of the job to not be an outlier (probability \(1 - \rho\)) and \(F_R(t)\):

\[
P(R < t) = (1 - \rho) F_R(t) = F_R(t)
\]

\(^1\)EGEE: http://www.eu-egee.org
\(^2\)knowARC: http://www.knowarc.org/
Conversely, the latency is longer than $t$ if the job is an outlier (probability $\rho$) or it is not an outlier (probability $1-\rho$) and it terminates before $t$ (probability $1 - F_R(t)$):

$$P(R > t) = \rho + (1-\rho)(1-F_R(t)) = 1 - (1-\rho)F_R(t) = 1 - \tilde{F}_R(t)$$

It has to be noted that although $P(R < t) = \tilde{F}_R(t)$, $F_R$ is not a cdf (it does not converge towards 1) and it cannot be considered as such in the subsequent calculations.

### 3.1 Experimental measurements

Our reference experimental data has been obtained on the biomed Virtual Organization (VO) of the EGEE production grid. With 80,000 CPU cores dispatched world-wide in more than 240 computing centers, EGEE represents an interesting case study as it exhibits highly variable and quickly evolving load patterns that depend on the concurrent activity of thousands of potential users. A given user has access only to a subset of the resources, as granted to her VO. When she submits jobs from her workstation, she uses an EGEE client known as a User Interface. A Workload Management Server receives and queues the jobs submitted before dispatching them to the connected computing centers. The gateway to each computing center is one or more Computing Element which hosts a batch manager to distribute the workload over the center computing resources. Different batch systems are operated in different centers. Each site is configured independently with site-specific policies determining the number of queues available and queues maximal wall-clock times.

### 3.2 Latency measures

Latency measures were collected by submitting a very large number of probe jobs. These jobs, consisting in the execution of an almost null duration `/bin/hostname` command, are only impacted by the grid latency. In the remainder we assume that the job execution time is known and we only focus on the grid latency that can significantly vary between different runs of a same computation task. To avoid variations of the system load due to monitoring, a constant number of probes was maintained inside the system: a new probe was submitted each time another one completed. For each probe job, the job submission date, the job final status and the total duration were logged. The probe jobs were assigned a fixed 10,000 seconds timeout beyond which they were considered as outliers and canceled. This value is far greater than the average latency observed ($\approx 500$ seconds).

Twelve traces sets collected at different times of the year and with different duration are exploited in this paper [7, 11]. They gather 10,893 probe jobs. A first trace acquired in September 2006 is denoted 2006-IX. The 11 other traces, acquired from end of year 2007 to beginning of year 2008 over one week each are denoted “year-week number”. In the future, we plan to make more systematic experiments by exploiting logs of the Grid Observatory\(^3\) that was recently set up. It gives practical foundations to the investigations conducted in this paper and it is a key to the systematic implementation of our methods in real applications.

### 3.3 Evaluation criteria

The submission strategies explored in the following are evaluated against two kinds of metric: the jobs performance (user point of view) and the number of submissions needed to achieve it (infrastructure point of view). The performance is assessed through the average latency time. In our probabilistic framework, an expression of the latency expectation (and its standard deviation) can be developed and then estimated from the traces mentioned above. This gives little insight when considering individual jobs but it makes perfect sense when considering applications involving a large number of jobs. The infrastructure load is assessed through the average number of jobs that is needed to achieve a given performance. Increasing the number of jobs in the system may impact the latency at some point. In our framework, we assume that the additional jobs have no measurable impact on the grid workload given its size and number of jobs processed ($\geq 150$ jobs / day on EGEE). As it will be demonstrated, this assumption makes sense as it is possible to achieve significant performance improvements for a small average number of additional jobs.

### 4. SINGLE RESUBMISSION

When facing high and highly variable latencies, a simple resubmission strategy consists in waiting until a timeout value $t_{\infty}$ and then canceling the job and resubmitting it, iterating this strategy until one job completes before $t_{\infty}$. The modeling of the total latency $J$, including resubmissions, has already been studied in [8] and its expectation can be expressed as a function of individual jobs latency ($R$) and the timeout value ($t_{\infty}$):

$$E_J(t_{\infty}) = \frac{1}{F_R(t_{\infty})} \int_0^{t_{\infty}} (1 - F_R(u)) du$$  \hspace{1cm} (1)

Minimizing this equation with respect to $t_{\infty}$ gives the optimal timeout value.

The standard deviation $\sigma_J$ is computed using the fact that

\[^3\text{http://www.grid-observatory.org/}\]
Table 1: Mean and standard variation of latency (R) and latency including resubmissions (J). The column “mean < 10^5” corresponds to the mean of latencies lower than 10,000 seconds. The column “mean with 10^5” is a low bound of the actual latency mean considering that latencies greater than 10,000s equal 10,000s.

<table>
<thead>
<tr>
<th>week number</th>
<th>mean &lt; 10^5s</th>
<th>mean with 10^5s</th>
<th>E_J</th>
<th>σ_R</th>
<th>σ_J</th>
<th>Δσ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-IX</td>
<td>570s</td>
<td>1042s</td>
<td>471s</td>
<td>886s</td>
<td>331s</td>
<td>-63%</td>
</tr>
<tr>
<td>2007/08</td>
<td>409s</td>
<td>2089s</td>
<td>500s</td>
<td>723s</td>
<td>358s</td>
<td>-51%</td>
</tr>
<tr>
<td>2007-36</td>
<td>446s</td>
<td>2739s</td>
<td>510s</td>
<td>748s</td>
<td>370s</td>
<td>-51%</td>
</tr>
<tr>
<td>2007-37</td>
<td>506s</td>
<td>3630s</td>
<td>617s</td>
<td>848s</td>
<td>486s</td>
<td>-43%</td>
</tr>
<tr>
<td>2007-38</td>
<td>447s</td>
<td>2739s</td>
<td>531s</td>
<td>682s</td>
<td>390s</td>
<td>-42%</td>
</tr>
<tr>
<td>2007-39</td>
<td>489s</td>
<td>3533s</td>
<td>596s</td>
<td>741s</td>
<td>482s</td>
<td>-35%</td>
</tr>
<tr>
<td>2007-50</td>
<td>660s</td>
<td>2341s</td>
<td>628s</td>
<td>1046s</td>
<td>475s</td>
<td>-55%</td>
</tr>
<tr>
<td>2007-51</td>
<td>478s</td>
<td>1716s</td>
<td>517s</td>
<td>510s</td>
<td>353s</td>
<td>-31%</td>
</tr>
<tr>
<td>2007-52</td>
<td>443s</td>
<td>1685s</td>
<td>476s</td>
<td>582s</td>
<td>334s</td>
<td>-43%</td>
</tr>
<tr>
<td>2007-53</td>
<td>449s</td>
<td>1977s</td>
<td>482s</td>
<td>678s</td>
<td>330s</td>
<td>-51%</td>
</tr>
<tr>
<td>2008-01</td>
<td>434s</td>
<td>1678s</td>
<td>499s</td>
<td>3175s</td>
<td>339s</td>
<td>+07%</td>
</tr>
<tr>
<td>2008-02</td>
<td>418s</td>
<td>1568s</td>
<td>441s</td>
<td>547s</td>
<td>278s</td>
<td>-49%</td>
</tr>
<tr>
<td>2008-03</td>
<td>538s</td>
<td>1484s</td>
<td>419s</td>
<td>1196s</td>
<td>269s</td>
<td>-78%</td>
</tr>
</tbody>
</table>

Figure 2: Expectation of execution time for different collection of b jobs, with respect to the timeout value.

We are now interested in the minimal execution time of the b parallel submissions. The probability of a job to finish before t is given by $F_R(t)$. Thus, the probability of all the b jobs to finish after t is given by $(1 - F_R(t))^b$. The probability of having at least one job running before t is thus given by $1 - (1 - F_R(t))^b$.

The new expected execution time can then be computed from equation 1 by replacing $F_R$ by $1 - (1 - F_R(t))^b$:

$$E_J(t_{\infty}) = \frac{1}{1 - (1 - F_R(t_{\infty}))^b} \int_0^{t_{\infty}} (1 - F_R(u))^b \, du$$

(3)

and the standard deviation:

$$\sigma^2_J(t_{\infty}) = \frac{2}{(1 - (1 - F_R(t_{\infty}))^b)^2} \int_0^{t_{\infty}} u(1 - F_R(u))^b \, du$$

$$+ \frac{2}{(1 - (1 - F_R(t_{\infty}))^b)^2} \int_0^{t_{\infty}} (1 - F_R(u))^b \, du$$

$$- \frac{1}{(1 - (1 - F_R(t_{\infty}))^b)^2} \left[ \int_0^{t_{\infty}} (1 - F_R(u))^b \, du \right]^2$$

(4)

5. MULTIPLE SUBMISSIONS

In order to further improve performance and to reduce chance of failure, multiple submission is often considered. A multiple submission strategy can easily be implemented within the EGEE Workload Management System (WMS) through burst submissions: for each job to be executed, a collection of b copies of this job is submitted. As soon as one job of the collection is running, all the other ones are canceled. If none of the jobs starts executing before the timeout value ($t_{\infty}$), the whole collection is canceled and resubmitted.
Table 2: Different values of the number of jobs in the collection \((b)\) leads to different values of optimal timeout and best expectation of execution time. A significant speed-up is achieved by the multi-submission strategy, even for low values of \(b\).

measure the improvement of \(E_J\) with one unit of \(b\). \(E_J\) is decreasing, significantly faster for smaller values of \(b\) than greater ones. This result is intuitive: adding one job to the collection has much more impact if the collection is very small than if it already contains many jobs.

In figure 3, the optimal values of \(E_J\) and associated standard-deviation \(\sigma_J\) are plotted for different periods of time with respect to the number of jobs in parallel. The decreasing curves confirm the previous observations.

6. DELAYED RESUBMISSION STRATEGY

The multiple submission strategy is efficient but aggressive for the infrastructure. An alternate delayed resubmission strategy, derived from the single resubmission is presented here. As illustrated in figure 4, it consists in submitting a single job, waiting until \(t_0\) and then, if it is not running yet, launching a copy of this job without canceling the first one before \(t_\infty\), and iterating this process until one job is running.

In order not to have more than 2 identical jobs in the system at the same time, we impose \(0 < t_0 < t_\infty\) and \((t_\infty - t_0) < t_0\) (this ensures that job 1 is canceled before job 3 is submitted, as illustrated on figure 4). The probability for a single job to timeout is given by \(q = 1 - F_R(t_\infty)\). If a job starts running at time \(t\) in the interval \([nt_0, (n - 1)t_0 + t_\infty]\) \((I_0\) on figure 4), this means that exactly \((n - 1)\) jobs have timed-out (probability \(q^{n-1}\)) and that either latency of job \(n\) is between \(t_0\) and \((t - (n - 1)t_0)\) (probability \(F_R(t - (n - 1)t_0) - F_R(t_0))\) or latency of job \(n + 1\) is lower than \((t - nt_0)\) (probability \(F_R(t - nt_0)\)). Since these last two events may both occur, the probability that at least one of them occurs is equal to the probability of their union minus the probability of their intersection.

![Figure 3: Evolution of minimal values of \(E_J\) and the associated \(\sigma_J\) values with respect to the number of jobs in the multi-submission \((b)\). Each curve corresponds to a set of data.](image)

![Figure 4: Principle of the delayed resubmission strategy: a single job is first submitted. At \(t_0\), if the job has not started, a copy of this job is submitted. If the first job is still not completed at \(t_\infty\), it is canceled. This strategy is iterated until a job is completed.](image)
the probability of their intersection, i.e.:

\[ F_J(t) = P(J < t|t \in [nt_0, (n+1)t_0 + t_\infty] ) \]
\[ = P(J < nt_0) + (1 - F_R(t_\infty))^{n-1} \]
\[ + \left( \bar{F}_R(t - (n-1)t_0) - F_R((n+1)t_0 - t_\infty) \right) \]
\[ - \left( \bar{F}_R(t - (n-1)t_0) - F_R(t_\infty) \right) F_R(t - nt_0) \]

Otherwise, if a job starts running at time \( t \) in the interval \([(n-1)t_0 + t_\infty, (n+1)t_0] \) (Fig. 4), this means that exactly \( n \) jobs have timeouts (with probability \( q^n \)) and that the latency of job \((n+1)\) is lower than \((t - nt_0)\) (probability \( F_R(t - nt_0) \)).

\[ F_J(t) = P(J < t|t \in [(n-1)t_0 + t_\infty, (n+1)t_0]) \]
\[ = P(J < (n-1)t_0 + t_\infty) + (1 - F_R(t_\infty))^n F_R(t - nt_0) \]

Deriving these last 2 equations leads to:

\[
\begin{align*}
    f_J(t) &= q^{n-1} \left( F_R(t - (n-1)t_0) ight) \\
    + (1 + F_R(t_0)) F_R(t - nt_0) \\
    - f_R(t - (n-1)t_0) f_R(t - nt_0) \\
    \forall t \in [(n-1)t_0 + t_\infty, (n+1)t_0] \\
\end{align*}
\]

Finally, by integration, and replacing the integration variable \( t \) by \( u = t - nt_0 \) or \( v = t - (n-1)t_0 \), we get:

\[ E_J = \int_{t_0}^{t_\infty} f_J(t)dt \\
= \int_{t_0}^{t_\infty} f_J(t)dt \\
+ \sum_{n=1}^{\infty} \left( \int_{t_0}^{(n-1)t_0 + t_\infty} f_J(t)dt + \int_{(n-1)t_0 + t_\infty}^{(n+1)t_0} f_J(t)dt \right) \\
= \int_{t_0}^{t_\infty} f_R(t)dt + \sum_{n=1}^{\infty} \int_{t_0}^{(n-1)t_0 + t_\infty} (v + (n-1)t_0) f_R(v)dv \\
+ \left( 1 + F_R(t_0) \right) \sum_{n=1}^{\infty} \int_{(n-1)t_0 + t_\infty}^{t_\infty} (u + nt_0) f_R(u)du \\
+ \sum_{n=1}^{\infty} \int_{t_0}^{t_\infty} (u + nt_0) f_R(u)du \\
- \sum_{n=1}^{\infty} \int_{t_0}^{t_\infty} (u + nt_0) f_R(u)du \\
\]

Grouping terms and replacing the integer series by their values leads to:

\[ E_J = \int_{t_0}^{t_\infty} f_R(t)dt \\
+ \frac{1}{1 - q} \int_{t_0}^{t_\infty} v f_R(v)dv + \frac{qt_0}{(1 - q)^2} (F_R(t_\infty) - F_R(t_0)) \\
+ \frac{1}{1 - q} \int_{t_0}^{t_\infty} u f_R(u)du \\
+ \left( t_0 + F_R(t_0) \right) F_R(t_\infty - t_0) \\
+ \frac{qt_0}{(1 - q)^2} (F_R(t_\infty) - \bar{F}_R(t_\infty - t_0)) \\
- \left( t_0 + F_R(t_0) \right) f_R(u + t_0) f_R(u)du \\
- \frac{1}{1 - q} \int_{t_0}^{t_\infty} u f_R(u + t_0) f_R(u)du \\
\]

This expression has to be minimized numerically with respect to \( t_0 \) and \( t_\infty \). In figure 5 the surface profile of \( E_J \) has been computed using dataset 2006-IX. This dataset leads to a best \( t_0 \) value of 339s, a best \( t_\infty \) value of 485s and a minimum value for the expected execution time of 431s, which is smaller than the single resubmission strategy but higher than the multiple resubmission strategy for \( b \geq 2 \). Although minimizing \( E_J \) leads to the best performance from a user point of view, it might also load the infrastructure by increasing the number of redundant jobs, which we study in the following.

**6.1 Number of parallel jobs**

With the delayed resubmission strategy, a variable number of jobs may be running on the infrastructure at any time. Let \( N_{/j} \) denote the average number of jobs needed, computed as the normalized sum of the number of parallel jobs running at each instant. \( N_{/j} \) is to be compared with the value of \( b \) in the case of multiple resubmissions. Let \( l \) be the latency of a job with parameters \( t_0 \) and \( t_\infty \). We can notice that, after the first period \([0; t_0]\), the number of jobs in parallel is periodic with a \( t_0 \)-period of time. In the following, let \( n \) be an integer such that \( l \) is in \([nt_0, (n+1)t_0]\).
Three cases have to be distinguished:

n = 0; Then, obviously, \( N_{t_0} = 1 \).
n = 1; Two different cases have to be considered:

\( l < t_\infty \): only one job is running until \( t_0 \). Then, a second job is running in parallel with the first one during a period of \((l - t_0)\) which leads to:

\[
N_{t_0} = \frac{t_0 + 2(l - t_0)}{t} = 2 - \frac{t_0}{t}
\]

\( l \geq t_\infty \): only one job is running until \( t_0 \). Then, two jobs are running in parallel during a period of \((t_\infty - t_0)\).

After that, the first job is canceled and there will be only one job running during \((l - t_\infty)\) leading to:

\[
N_{t_0} = \frac{t_0 + 2(t_\infty - t_0) + (l - t_\infty)}{t}
\]

n > 1; For both cases detailed below, only one job is running until \( t_0 \). Then, for \((n - 1)\) period of \( t_0 \) time, we will have two jobs in parallel during \([0, (n-1)t_0)\) and only one job during \([t_0, (n-1)t_0)\).

\( l \in (nt_0; (n-1)t_0 + t_\infty) \): after the \((n-1)t_0\) first periods, two jobs are running in parallel during \((l - nt_0)\) which leads to:

\[
N_{t_0} = \frac{t_0 + (n-1)t_\infty + 2(l - nt_0)}{l}
\]

\( l \in ([n-1]t_0 + t_\infty; (n+1)t_0) \): after the \((n-1)t_0\) first periods, two jobs are running in parallel during \([t_0, (n-1)t_0 + t_\infty)\) and then only one job during \((l - ((n-1)t_0 + t_\infty))\) which leads to:

\[
N_{t_0} = \frac{t_0 + (n-1)t_\infty + 2(t_\infty - t_0) + (l - (n-1)t_0 - t_\infty)}{l}
\]

It can be verified that \( \lim_{n \rightarrow \infty} N_{t_0} = \frac{t_\infty}{t} \). However, considering realistic values of \( l \) and \( t_\infty \), \( n \) will usually be small. We can demonstrate that \( N_{t_0} \in [1; 2 - \frac{t_\infty}{t}] \), thus leading to a maximum of 1.5 in the case of \( n = 1 \). In the following, we will experiment different values of \( N_{t_0} \) in order to study the variations of the minimal value of \( l \).

### 6.2 Imposing the ratio \( \frac{t_\infty}{t} \)

Although this ratio can easily be imposed, it only corresponds to the asymptotic behavior of \( N_{t_0} \) and not to \( N_{t_0} \)'s actual value. \( N_{t_0} \) is thus computed in each case, using the minimal \( E_J \) values computed using equation 5.

In table 3, the results of the minimization of equation 5 for different values of the ratio \( \frac{t_\infty}{t} \) are presented, leading to different values of \( N_{t_0} \). The minimal values of \( E_J \) correspond either to \( n = 0 \) (\( E_J < t_0 \)), where \( N_{t_0} = 1 \), or to \( n = 1 \) (\( N_{t_0} \) between 1 and 1.45).

The minimal values of \( E_J \) are compared with the number of jobs in parallel on figure 6. The minimal value for the delayed resubmission strategy, obtained from a global minimization of equation 5, is \( E_J = 431s \) for a mean of 1.2 jobs in parallel. This minimal value is lower than the one obtained with the single resubmission strategy by 8.3%. However, we obtain a lower value with the multiple submission strategy with at least two jobs in parallel. To fairly compare different strategies, a measure of the cost induced by an increase of the mean number of jobs in parallel is needed.

<table>
<thead>
<tr>
<th>( \frac{t_\infty}{t} )</th>
<th>( N_{t_0} )</th>
<th>( \text{best } t_\infty )</th>
<th>( \text{best } t_0 )</th>
<th>( \min E_J )</th>
<th>( \Delta(100%) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>1</td>
<td>556s</td>
<td>505s</td>
<td>458s</td>
<td>-2.7%</td>
</tr>
<tr>
<td>1.2</td>
<td>1</td>
<td>556s</td>
<td>463s</td>
<td>447s</td>
<td>-5.0%</td>
</tr>
<tr>
<td>1.3</td>
<td>1.07</td>
<td>528s</td>
<td>406s</td>
<td>438s</td>
<td>-6.9%</td>
</tr>
<tr>
<td>1.4</td>
<td>1.18</td>
<td>496s</td>
<td>354s</td>
<td>432s</td>
<td>-8.2%</td>
</tr>
<tr>
<td>1.5</td>
<td>1.32</td>
<td>445s</td>
<td>297s</td>
<td>434s</td>
<td>-7.7%</td>
</tr>
<tr>
<td>1.6</td>
<td>1.37</td>
<td>435s</td>
<td>272s</td>
<td>444s</td>
<td>-5.6%</td>
</tr>
<tr>
<td>1.7</td>
<td>1.39</td>
<td>431s</td>
<td>254s</td>
<td>457s</td>
<td>-2.9%</td>
</tr>
<tr>
<td>1.8</td>
<td>1.41</td>
<td>426s</td>
<td>237s</td>
<td>462s</td>
<td>-1.9%</td>
</tr>
<tr>
<td>1.9</td>
<td>1.47</td>
<td>425s</td>
<td>221s</td>
<td>466s</td>
<td>-1%</td>
</tr>
<tr>
<td>2</td>
<td>1.45</td>
<td>423s</td>
<td>211s</td>
<td>469s</td>
<td>-0.5%</td>
</tr>
</tbody>
</table>

### Figure 6: Minimal expected execution time according to delayed resubmission strategy (plain curve) or multiple submission strategy (dashed curve), with respect to the mean number of job copies running in parallel.

### Figure 7: Multiple submission strategies can also reduce the global load of the infrastructure when they yield sufficient time gain. Top: the single resubmission strategy leads to an average number of jobs of 1 on \([0, T]\). Bottom: a multiple submission strategy reduces it to 0.5 on this time period.
Figure 8: $\Delta$ value according to 2 different strategies: delayed resubmission strategy or multiple submission strategy, with respect to the mean number of job copies running in parallel.

7. DISCUSSION ON THE STRATEGIES COST

Although submitting twice the same job increases the grid load, it still leads to a global benefit for the infrastructure if the gain in time is higher than 2. Indeed, in this case, the expectation of the number of jobs in the system decreases, as illustrated on figure 7. This idea can be extended to any number of jobs, trying to satisfy this relation:

$$E_J(\text{delayed sub. with } N_{ij}) < \frac{E_J(\text{single resub. with } b = 1)}{N_{ij}}$$

The cost of the strategy is then:

$$\Delta_{cost} = N_{ij} \cdot \frac{E_J(\text{with } N_{ij})}{E_J(\text{with } b = 1)}$$

for different job numbers.

Using this definition, the cost of the single resubmission strategy detailed in Section 4 is 1. In table 4, different samples of $N_{ij}$, $E_J$ and $\Delta_{cost}$ are displayed for the different strategies studied in this paper. Obviously, in the multiple submission strategy, we have $N_{ij} = b$. Figure 8 presents the values of $\Delta_{cost}$ with respect to $N_{ij}$. For integer values of $N_{ij}$ (corresponding to the multiple submission strategy), values of $\Delta_{cost}$ are increasing and greater than 1. For values of $N_{ij}$ smaller than 2, the values of $\Delta_{cost}$ starting from 1, are decreasing until a minimal value of 0.94, and then increasing beyond 1. All values of $\Delta_{cost}$ smaller than 1 indicate that the load on the grid is lower than a single running job (the strategy of single resubmission). Using the smallest value of $\Delta_{cost}$ leads to the smallest occupation of the grid while reducing the latency. The results that are displayed in table 4 are those, given a value of $\frac{t_{min}}{T_{max}}$, that minimize $E_J$. Considering the minimization with respect of the $\Delta_{cost}$ value, the minimum is reached for $\Delta_{cost} = 0.93$, $t_0 = 439s$ and $t_{\infty} = 579s$ leading to $E_J = 439s$ (less than $E_J(b=1) = 471s$). Moreover, a consequence of reducing the grid load might be a reduction of the latencies. This has not been studied in this paper and it is subject for future work.

7.1 Results on data from 2007-2008

Table 4: In the case of the delayed resubmission strategy, for each $\frac{t_{min}}{T_{max}}$ value, the minimal $E_J$ is computed. Corresponding values of $N_{ij}$ and $\Delta_{cost}$ are thus given. We observe that a ratio $\frac{t_{min}}{T_{max}}$ of 1.25 appears to be the optimal solution with respect to the $\Delta_{cost}$ value. For other strategies, minimal $E_J$ is computed from $N_{ij}$. For higher number of jobs, the cost is increasing. These results have been computed on the 2006-IX dataset.

Left part of table 5 presents, for each week of the dataset from 2007-2008, the minimum value of $\Delta_{cost}$ obtained using the delayed resubmission strategy. For the 5 first weeks, the minimal value of $\Delta_{cost}$ is higher than 1 while, for the other 6 weeks, including the whole period, $\Delta_{cost}$ presents a minimum less than 1, as it was the case for the dataset studied in the previous section. This shows that, depending on the grid workload, the delayed strategy is not always optimal in term of $\Delta_{cost}$: the single submission is to be chosen instead.

For each period of time, the values of $t_0$ and $t_{\infty}$ corresponding to the minimal value of $\Delta_{cost}$ are given. We have also tested the stability of such minimal values by adding variations up to ±5 seconds to each of $t_0$ and $t_{\infty}$. The study was limited to integer values of $t_0$ and $t_{\infty}$ because having higher precision of resubmission is not realistic in practice. The maximal values of $\Delta_{cost}$ and the maximum relative difference with respect to the minimum are given on right of table 5 for all minima less than 1. Some $\Delta_{cost}$ stay below 1 while others grow more than 1 but still below 1.1 where the highest relative difference is of 14%. This shows a relative stability that needs to be enforced by a good estimation of both optimals $t_0$ and $t_{\infty}$.

7.2 Practical implementation

Up to now, traces were studied a posteriori. However, exploiting the $\Delta_{cost}$ optimization strategy in practice requires to collect data for estimating $t_0$ and $t_{\infty}$ prior to the jobs execution. In the experiment reported in table 6, the variations of $\Delta_{cost}$ for the different values of $t_0$ and $t_{\infty}$ that have been obtained on each time period are studied. Assuming that $\Delta_{cost}$ was computed from the measurements collected during any of the periods considered at random, the value of $\Delta_{cost}$ is shown (the optimal value, corresponding to the
Table 5: Minimal $\Delta_{\text{cost}}$ values for the different periods with corresponding values of $t_0$, $t_\infty$ and $E_J$. All $E_J$ values are below the ones obtained with the single resubmission strategy (see table 1). For the cases where the minimum is lower than 1.0, variations of $\Delta_{\text{cost}}$ around optimal $t_0$ and $t_\infty$ values (radius 5): maximal value and relative difference with the maximal value.

studied period’s measurements, is underlined). These results show a maximal variation of 13% (mean variation of 9%). Assuming now that $\Delta_{\text{cost}}$ was computed from the measurements collected the week before, the last column in the table displays the variation due to the use of the $t_0$ and $t_\infty$ optimal values from the week before in the computation of $\Delta_{\text{cost}}$. In that case, the relative difference is never larger than 6% and when $\Delta_{\text{cost}}$ is higher than 1, it is at a precision of $10^{-3}$.

8. CONCLUSION

In this paper we have studied 3 different job submission strategies that users can adopt in order to reduce the latency they experience on production grids. For each of those strategies, we have established a model of the expectation of the total latency (including resubmission) and its standard deviation. We have shown that all these strategies reduce the latency. Moreover, the delayed resubmission strategy reduces the latency requiring less than 2 jobs in parallel while the multiple submission strategy gives the higher latency reduction but at a higher cost.

We have then proposed a cost criterion which characterizes the conditions under which it is possible to obtain both a latency smaller than in the case of single resubmission and fewer parallel jobs.

Future work will concern the validation of this approach with real applications. The impact of each strategy on grid-applications makespan can be measured and averaged to take into account evolutive experimental conditions. In a second step, the impact of all grid users exploiting the same strategy can be simulated in a controlled environment.

Acknowledgments

The authors would like to thank Pierre Bernhard and Roger Marlin for fruitful discussions. This work is partially funded by the French national research agency (ANR), NeuroLog

<table>
<thead>
<tr>
<th>week</th>
<th>opt. $t_0$</th>
<th>opt. $t_\infty$</th>
<th>$E_J$</th>
<th>max $\Delta_{\text{cost}}$</th>
<th>max $\Delta_{\text{cost}}$ %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-36</td>
<td>422</td>
<td>423</td>
<td>1.001</td>
<td>510</td>
<td>1.09</td>
</tr>
<tr>
<td>2007-37</td>
<td>421</td>
<td>422</td>
<td>1.000</td>
<td>616</td>
<td>1.04</td>
</tr>
<tr>
<td>2007-38</td>
<td>427</td>
<td>428</td>
<td>1.001</td>
<td>530</td>
<td>1.025</td>
</tr>
<tr>
<td>2007-39</td>
<td>435</td>
<td>436</td>
<td>1.001</td>
<td>595</td>
<td>1.025</td>
</tr>
<tr>
<td>2007-50</td>
<td>466</td>
<td>467</td>
<td>1.001</td>
<td>627</td>
<td>1.04</td>
</tr>
<tr>
<td>2007-51</td>
<td>499</td>
<td>662</td>
<td>0.954</td>
<td>494</td>
<td>1.09</td>
</tr>
<tr>
<td>2007-52</td>
<td>455</td>
<td>595</td>
<td>0.955</td>
<td>453</td>
<td>0.97</td>
</tr>
<tr>
<td>2007-53</td>
<td>463</td>
<td>613</td>
<td>0.961</td>
<td>463</td>
<td>0.97</td>
</tr>
<tr>
<td>2008-01</td>
<td>489</td>
<td>525</td>
<td>0.984</td>
<td>489</td>
<td>1.03</td>
</tr>
<tr>
<td>2008-02</td>
<td>420</td>
<td>575</td>
<td>0.953</td>
<td>420</td>
<td>1.09</td>
</tr>
<tr>
<td>2008-03</td>
<td>395</td>
<td>530</td>
<td>0.943</td>
<td>395</td>
<td>0.95</td>
</tr>
<tr>
<td>2007-08</td>
<td>481</td>
<td>635</td>
<td>0.963</td>
<td>481</td>
<td>1.09</td>
</tr>
</tbody>
</table>

Table 6: Variations of $E_J$ and $\Delta_{\text{cost}}$ for different values of $t_0$ and $t_\infty$ corresponding to optimal $\Delta_{\text{cost}}$ values for each week and the whole period 2007/08.
project\footnote{Neurolog: http://neurolog.polytech.unice.fr} under contract number ANR-06-TLOG-024 and CNRS ST2I PEPS “Grid Observatory”. We are grateful to the EGEE European project for providing the grid infrastructure and user assistance.

9. REFERENCES


