Multi-scale Real-time Grid Monitoring with Job Stream Mining

Xiangliang Zhang Michèle Sebag Cécile Germain-Renaud
TAO – INRIA – CNRS
Université Paris-Sud 11, F-91405 Orsay, France
Email: {xlzhang, sebag, cecile}@lri.fr

***Abstract*—The ever increasing scale and complexity of large computational systems ask for sophisticated management tools, paving the way toward Autonomic Computing. A first step toward Autonomic Grids is presented in this paper; the interactions between the grid middleware and the stream of computational queries are modeled using statistical learning. The approach is implemented and validated in the context of the EGEE grid. The GSTRAP system, embedding the STRAP Data Streaming algorithm, provides manageable and understandable views of the computational workload based on gLite reporting services. An online monitoring module shows the instant distribution of the jobs in real-time and its dynamics, enabling anomaly detection. An offline monitoring module provides the administrator with a consolidated view of the workload, enabling the visual inspection of its long-term trends.**

I. INTRODUCTION

The ever increasing grid resources and computational load likewise increase the demands on the administrator, in charge of monitoring the grid status and maintaining the job running process. More generally, the increasing pressure on system administrators paved the way for an emerging field, Autonomic Computing (AC) [1]. AC aims at providing system administrators with scalable and high level management tools, with the long term goal of self-configuring, self-healing, self-protecting and self-optimizing large computing systems [2].

This paper is more specifically interested in Autonomic Computing facilities for the EGEE (*Enabling Grid for E-SciencE*[1]) grid. It is one of the largest multi-disciplinary grid infrastructures in the world, developed in the European Community Infrastructure Framework. EGEE has been built to address e-Science computational needs (in e.g., high energy physics, life sciences, computational chemistry, financial simulation) through an efficient distributed computing system, enabling the communalization of resources. Computational experiments in e-Science require high CPU, large memory and huge storage capacities. EGEE currently involves 100,000 CPUs with 20 Petabytes of storage. These resources are integrated within the gLite middleware [3], and EGEE currently supports up to 300,000 jobs per day on a 24×7 basis.

The goal of this paper is to provide the administrator with some facilities for inspecting the flow of jobs submitted to EGEE, and the grid reactions. The complex interactions between the grid middleware and the actual computational queries can hardly be modeled using first-principle based approaches, at least with regard to the desired level of precision. Therefore, an empirical approach will be investigated, exploiting the gLite reports on the lifecycle of the jobs and on the behavior of the middleware components. Actually, gLite involves extensive monitoring facilities, generating a wealth of trace data; these traces include every detail about the internal processing of the jobs and functioning of the grid. How to turn these traces in manageable, understandable and valuable summaries or models is acknowledged to be a key operational issue [4].

This paper presents an algorithm called GSTRAP achieving a compact online and offline modeling of the gLite trace data. The main GSTRAP originality compared with the state of the art in Grid Monitoring (section II-A) is to focus on the interpretation of the data, relying on Data Streaming principles. Data Streaming, a Data Mining field (section II-B), is concerned with the online exploitation of large data flows such as generated in the realm of telecommunications, Web queries, or sensor networks. A Data Streaming algorithm first presented in [5], STRAP, is extended to form the GSTRAP algorithm which provides multi-scale analysis facilities to the grid administrator.

Online, STRAP models the current trace through a set of representative jobs[2] referred to as exemplars. STRAP seamlessly compares each new job with the exemplars; if statistically close to an exemplar, the new job is used to update the exemplar representativity; otherwise, the job is considered to be an outlier and stored in a reservoir. The critical issue in Data Streaming is to preserve an efficient tradeoff between robustness (discarding outliers) and sensitivity (quickly detecting and catching up the changes in the underlying data distribution). In STRAP, this tradeoff is handled through a statistical change detection test, the Page-Hinkley (PH) test [6], [7]. Upon the PH test triggering, the model is rebuilt from the current exemplars and the most recent outliers.

The online gLite summary thus is made of a set of actual jobs together with their current representativity, enabling the administrator e.g., to inspect the percentage of jobs that are successfully completed with high waiting time, and the

[1] http://www.eu-egee.org/

[2] More precisely, a job is meant as a summary of the job lifecycle in EGEE; the term *job* is used for the sake of simplicity.

percentage of jobs with various types of errors. Furthermore, the model rebuilding sequence indicates how frequently the underlying distribution was considered to be changed, providing insights into the dynamics of the grid usage.

The offline summary is provided by GSTRAP, exploiting the set of exemplars constructed by STRAP in the last months. Super-exemplars are built from the exemplars and used to visualize the grid dynamics *a posteriori* in form of a tapestry (Fig. 8 and 9), showing the long-run trends in the EGEE traffic, and any potential regularity in the peaks of usage.

GSTRAP has been empirically validated on a 5-million job trace, the 5-month log from 39 EGEE Resource Brokers. On the algorithmic side, the validation criteria regard the computational scalability, the stability and accuracy of the model (e.g. whether the jobs associated to a given exemplar have the same label, successfully finished or failed with various operational failure codes). On the applicative side, online modeling is shown to enable the instant detection of regime drifts (e.g., clogging of LogMonitor); offline modeling gives further insights on repetitive abnormal behaviors.

The paper is organized as follows. Section II briefly reviews and discusses related work in grid monitoring, and presents the state of the art in data streaming. Section III gives an overview of the GSTRAP algorithm. The experimental setting is presented in section IV. Section V reports on the experimental results. The paper concludes with a discussion and some perspectives for further research.

II. STATE OF THE ART

This section reviews some related work in Grid Monitoring, and provides a short introduction to Data Streaming.

A. Grid Monitoring

Grid Monitoring involves two main functionalities, respectively acquisition and usage of the relevant information. Acquisition includes sensors that instrument grid services or applications, and data collection services that filter, centralize and/or distribute the sensor data to the usage functionality. Acquisition raises challenging scalability and implementation issues. A plethora of architectures have been proposed and deployed. They provide a distributed information management service supporting in principle any kind of sensors. In the EGEE framework, deployed architectures include R-GMA [8], Ganglia, Nagios [9], MonALISA [10], gridIce [11], and SCALEA-G [12]. They aim at job lifecycles (e.g., Job Provenance [13] and the gLite Logging and Bookkeeping service [3]) or service and machine availability (e.g., SAM [14], Lemon [15] and GMS [16]).

Usage, which is more specifically investigated in this paper, includes consumer services such as real-time presentation and interpretation. It also includes middleware services as far as feedback loops are considered, typically in the Autonomic Computing framework. Many architectures and integration frameworks such as the EGEE DASHBOARD [17] and Real Time Monitor [18] also offer advanced presentation, user interaction and reporting facilities, although no interpretation facility is provided. Some of them include a software infrastructure for plugging analysis and feedback tools.

Actually, data interpretation, meant as revealing meaningful (compound) features which go beyond elementary statistics, is much less developed in the grid area. Ontologies about the grid behavior have been proposed [19], [20], including a taxonomy of relevant concepts and the structuration thereof. Although specified in OWL, these ontologies however provide limited inference capacities at the moment, Interpretation-wise, Grid Monitoring mostly focuses on feeding schedulers with educated guesses, e.g. the prediction of the upcoming workload. The well-known Network Weather Service [21] has pioneered a supervised learning-based approach, extracting the parameters of various elementary predictors and combining them in the spirit of boosting. [22] likewise predicts the estimated response time (with a given confidence interval) for batch-scheduled parallel machines. [23] categorizes load models on shared clusters. The integration of ontologies, monitoring and grid scheduling of work flows has been explored in the Askalon project [24], [25].

With regard to fault detection and diagnosis, the main EGEE tool is *Service Availability Monitoring* (SAM) [14]. Following an end-to-end probing strategy, SAM proceeds by sending commands, transactions, or service requests from highly reliable machines, and analyzing their results online. The critical issue for probing strategies is to define an adaptive and hierarchical probing scheme [1]. Periodically running a fixed scheme, which is the pre-planned scheme approach in SAM for instance, is both hardly efficient[3] and intrusive.

[26] and [27] propose more adaptive and thus less intrusive methods for detecting misbehaving users in volunteer computing grids. Interestingly, [26] is based on sequential testing, closely related to the change detection used in GSTRAP (section III-A).

An alternative to probing strategies, referred to as passive monitoring, exploits the actual traces of the grid activity. Assuming that the vast majority of grid components are fully operational and that faulty components have significant impact on the execution, [16] detects ill-configured computing elements (CE) as outliers with respect to the distribution of the CE behaviors, using distributed data mining algorithms. An elementary usage functionality, implemented in gLite, is to simply *blacklisting* the sites reporting a high failure rate.

B. Data Streaming

This paper extends the above mentioned passive monitoring approaches through Data Streaming. Let us introduce briefly the field of Data Streaming, referring the interested reader to [28], [29] for a more comprehensive presentation.

While Data Mining is concerned with extracting knowledge from (very) large databases, Data Streaming is specifically concerned with data flows, typically related to complex production systems such as telecommunications, Internet or

[3]While the diagnosis is nothing near instantaneous, only basic functionalities are checked.

electrical power networks [29]. In the general case, data flows raise two additional issues compared with standard databases. Firstly, the data acquisition rate is huge such that each data item can be looked at only once; linear computational complexity (up to logarithmic terms) is a must. Secondly and most importantly, the data flow cannot be considered as generated from a stationary distribution; there are changes in the system under study or its environment. Data Streaming thus requires specific algorithms, able to smoothly incorporate new information and follow the changes in the flow distribution, through robustly filtering out the outliers and forgetting outdated information [28]. The main challenge raised by the flow dynamics is to tell outliers from the first items generated by a new distribution component. Among the main applicative goals of Data Streaming are the classification of data items into classes (supervised learning), their categorization in clusters (unsupervised learning), the detection of anomalies and extreme events (e.g., intrusion detection), and the detection of correlations in chronic data. Most above goals rely on modeling the data distribution, and maintaining this model in real-time.

Both scalability and adaptability issues are indeed relevant to mining the gLite traces, characterizing an average 300,000 jobs per day (on the increase) generated by a huge variable community of users in a variable environment. The goal is to extract an *understandable* model of the job distribution (unsupervised learning; see discussion in section IV).

III. Grid monitoring with GStrAP

This section describes the GStrAP system. Its overall architecture (Fig. 1) includes an acquisition module and two analysis modules besides the Data Streaming StrAP algorithm, first presented in [5] and briefly summarized thereafter for the sake of self-containedness.

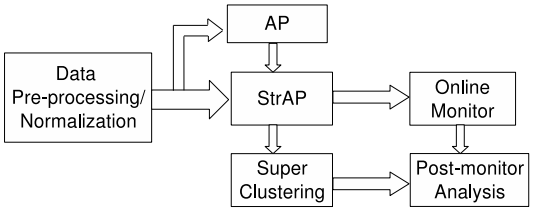


Fig. 1. Framework of Grid-adapted StrAP

A. The StrAP algorithm

Clustering algorithms can be described as follows. Letting $\mathcal{E}$ denote a set of items $\{x_1, \ldots, x_N\}$, a clustering algorithm aims at partitioning $\mathcal{E}$ into (usually) disjoint subsets, called clusters, such that items in a same cluster are "similar" and items in different clusters are "dissimilar" in the sense of a given distance or dissimilarity function.

Affinity Propagation (AP), a clustering algorithm proposed by Frey and Dueck in 2007 [30], features two desirable properties. Firstly, AP builds interpretable clusters, each cluster being represented by an actual item referred to as exemplar; secondly, contrasting with K-centers (which also builds exemplar-based clusters), AP enforces the quality and stability of the clustering solution. Formally, AP associates to each item x_i an exemplar noted $x_{\sigma(i)}$; the sum of the square dissimilarities $d^2(x_i, x_{\sigma(i)})$ over all items, referred to as *distortion*, is minimized using a message passing algorithm. AP involves a single parameter s^*; the number K of clusters is governed by the penalty s^* paid for incrementing the number of exemplars. The price to pay for these understandability and stability properties is a quadratic computational complexity ($\mathcal{O}(N^2)$ up to logarithmic terms, being N the number of items), practically forbidding its use on large scale applications.

The first extension, Weighted AP (WAP) was thus proposed to extend AP to the case of weighted or multiply-defined items with no loss of generality. WAP was thereafter embedded into a hierarchical Divide and Conquer approach, splitting the datasets into subsets and iteratively clustering the exemplars extracted from the subsets, thereby reducing the computational complexity to $\mathcal{O}(N^{1+\varepsilon})$ (see [5] for more detail).

The second extension, StrAP was proposed to deal with data flows using a four step process (Alg. 1):
1. Initialization: the model (first set of exemplars) is extracted from the first bunch of items.
2. Loop: each item is checked against the current exemplars. If it is statistically different (outlier), it is put into the *reservoir*, otherwise the statistics of the model (the nearest exemplar) are updated accordingly.
3. Change step: a statistical change-point-detection (Page-Hinkley [6], [7]) test on the outlier frequency is used to detect whether new clusters are emerging.
4. Rebuild step: upon triggering the PH test, the model is rebuilt from the current exemplars and the reservoir using WAP.

Algorithm 1 Grid-adapted StrAP Algorithm

```
Job stream x_1, ... x_t, ...; parameters ε, Δ
Init
  AP(x_1, ..., x_T) → Model                          StrAP
  Reservoir = {}
for t > T do
  if Outlier(Model, x_t, ε) then
    Reservoir ← x_t
  else
    Update StrAP model
  end if

  if Restart                          (Page-Hinkley test) then
    Model = Rebuild (Model, Reservoir, Δ)
    Reservoir = {}
  end if
end for
```

StrAP is parameterized from the outlier threshold ε, and the memory parameter Δ: exemplars which have not been visited for a given time period Δ are discarded in the rebuild

step.

B. The Acquisition Module

The acquisition module, interfaced with either the Real Time Monitor system (RTM) [18] or standalone files (e.g. for validation purposes), provides the description of the jobs through XML records. The RTM system, fully operational since the beginnings of EGEE, meets the real-time acquisition constraints; so does STRAP.

Each XML record provides a summary of the job lifecycle, as depicted in Fig. 2. A job is first *submitted*, then *waiting* for the Workload Management System (WMS) to find a matching resource. Once a resource is found, the job is *ready* for transfer, then transferred to the resource and *scheduled*, i.e. enqueued in the local batch system. When selected, the job is *running*, until being successfully ended (*done OK*), or failed (*done failed*). Notably, the resource allocation is never reconsidered after the matching step; upon failure, the job is resubmitted and goes through the whole process one more time. Early termination (*aborts*, *cancels*) triggered by either the user or the middleware components can occur at any step in the job lifecycle.

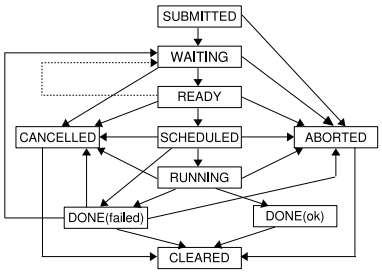


Fig. 2. Life cycle of jobs submitted to Grid

Formally, the timestamp of each event (transition in the graph, Fig. 2) is recorded by the Logging and Bookkeeping (L&B) service, i.e. the gLite information system relevant to active jobs. Each job will thereafter be summarized from the following six attributes (time durations computed from the timestamps), centered and normalized:

- Submission_Time: time between job registration and transfer to WMS;
- Waiting_Time: time to find a matching resource;
- Ready_for_Transfer_Time: acceptation and transfer delays (*waiting* + *ready* time), as reported by the JobController (JC);
- Ready_for_CE_accept_Time: the same as Ready_for_Transfer_Time, but as reported by the LogMonitor (LM)[4];
- Scheduled_Time: queuing delay;
- Running_Time: execution time.

Upon a job failure at any point in the lifecycle, the subsequent services are not reached and the reported durations are set to 0.

[4]JC is a standalone logging service while the LM integrates various logs in the Logging and Bookkeeping database. Although the third and fourth attributes are redundant in principle, their discrepancies provide valuable information as will be seen in section V.

To account for this fact and disambiguate the data description, six additional boolean attributes are considered, indicating whether the job has reached the corresponding steps of the lifecycle.

C. Analysis Modules

The online and offline analysis modules provide the administrator with multi-scale views of the job flow. The first module, aimed at online monitoring, firstly displays the instant distribution of the jobs (Fig. 7; see next section). The model rebuilding sequence (Fig. 4) indicates how fast the job dynamics is changing and the amplitude of the variations.

The second module aims at offline monitoring. All exemplars built by STRAP over a large period of time are considered and used by AP to extract "super-exemplars". The latter provide a common description visual representation of the long-term regularities and trends in the job distribution (Fig. 8,9).

IV. EXPERIMENT GOAL AND SETTINGS

GSTRAP has been empirically validated on the gLite-operated jobs processed by the EGEE grid from 2006-01-01 to 2006-05-31. The job flow includes 5,268,564 jobs in total, each one being labeled as *good* (successfully finished) or after its error type (e.g. *Cancel requested by WM*, *RB Cannot plan*). The temporal distribution of the jobs is depicted on Fig. 3.

45 types of errors are represented in the job flow, among which 20 are frequent (more than 1,500 occurrences). The grid experts nevertheless advocated for an unsupervised learning approach (the labels are not used in the clustering process), arguing that the error types cannot be directly interpreted in terms of failure. For instance, while *RB cannot plan* means that WMS was unable to find a matching resource, the cause might be that WMS was unaware of released resources, or that the the user's requests were truly impossible to satisfy.

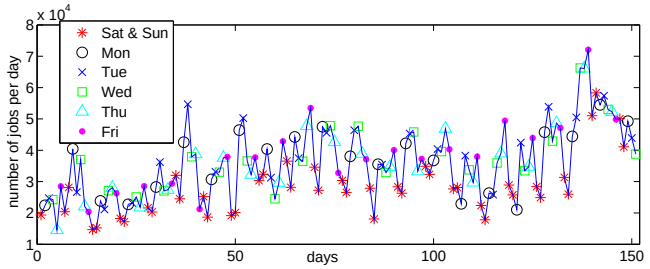


Fig. 3. EGEE load: number of jobs ($\times 10^4$) per day

The goal of the experiments regards both the algorithmic performance of GSTRAP (scalability w.r.t. the time constraints; quality of the clusters w.r.t. the error classes, see below), and the usability of the multi-scale views yielded by GSTRAP.

All reported results have been obtained as follows. The distance used by STRAP is the Euclidean distance in $\mathbb{R}^{12}$ (remind that all numerical attributes are centered and normalized). The stream model is initialized from the first 1,000 jobs

(initial subset); the s^* parameter is set to the median value of the distance between any two jobs in the initial subset. The outlier threshold ε is the average distance between an item and the associated exemplar in the initial model. The parameters of the PH test are $\lambda = 50$ and $\delta = .01$; whereas the latter parameter takes a standard value, further research is concerned with online adjustment of λ (see section VI). The time parameter Δ is set to 10,000, with a minimum number of queries per day of 15,000.

V. Experimental Validations

This section first reports on the algorithmic performance of GSTRAP. Its scalability (10,000 jobs per minute with Matlab, 40,000 jobs per minute for C/C++) meets the real-time constraints[5]. The quality of the clusters with respect to the classes of errors in the job stream is described and analyzed. The applicative performance, and the interpretation of the online and offline monitoring results, are thereafter discussed.

A. Dynamics of the Job Stream

Fig. 4 displays a summary of the model rebuilding sequence, indicating the number of times the stream model was rebuilt per day. Interestingly, the dynamics of the stream is not correlated with its volume (Fig. 3); the high number of model rebuilding on some days is caused by the apparition of brand new patterns; on other days, the same patterns keep disappearing and reappearing. Further study is on-going to explain this phenomenon.

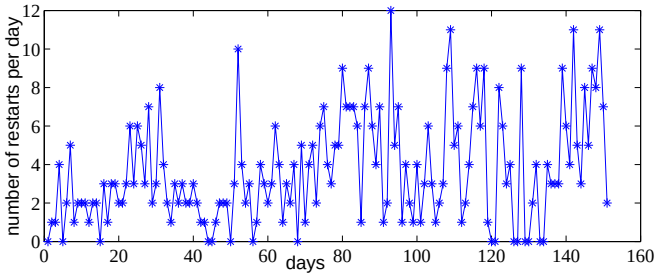


Fig. 4. GSTRAP: Number of model rebuilding per day

B. Clustering Quality

Despite the reservations made by the experts about the interpretation of the labels (section IV), the quality of clusters is commonly measured in terms of supervised learning, by comparing the label of the items and the label of the cluster (the label of the exemplar). Two measures will be considered. The *accuracy* is defined as the percentage of jobs which have the same label as the exemplar they are associated to. The *purity* is the average, over all clusters, of the cluster accuracy, defined as the percentage of jobs in the cluster with the same label as the majority of jobs in the cluster. The latter measure is less prone to be biased than the former one, in case of imbalanced clusters and classes.

[5]The experiments are run on an Intel 2.66GHz Dual-Core PC with 2 GB memory. GSTRAP processing rate is actually well above the current capacities of the gLite middleware regarding the job injection in the system.

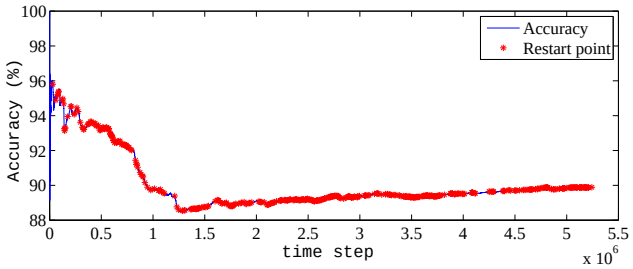


Fig. 5. GSTRAP: Clustering accuracy along time

Experimentally, the clustering accuracy is over 90% for the first 300,000 jobs; it is consistently above 88% during the whole period and slowly increases along time (Fig. 5). The in-depth analysis of the misclassified jobs shows four main cases of misclassification. The first three associate jobs with label *"Aborted by user"*, *"Job proxy is expired"*, or *"Job RetryCount (3) hit"* to exemplars with label *"Job RetryCount (0) hit"*. Actually, the last error label (*"Job RetryCount (0) hit"*) is a consequence of other types of error: it is triggered by the job resubmission when the job failed earlier (due to some unknown types of error), which might explain why the clustering process does not allow for clearly separating this class from the others. The fourth misclassification case associate jobs with label *"Successfully Finished"* to exemplars with label *"Job proxy is expired"*.

The purity, together with the number of clusters, is depicted in Fig. 6. The high number of clusters (between 50 and 250) is to be taken relatively to the number of jobs per day ($> 15,000$, Fig. 3). Surprisingly, the average clustering purity is consistently higher than 90% and much better than the clustering accuracy, despite the fact that purity is known to be a more pessimistic measure than accuracy in general. This is explained as most misclassified jobs belong to large clusters (*"Job RetryCount (0) hit"* or *"Job proxy is expired"*); the vast majority of clusters related to rare error types are accurate.

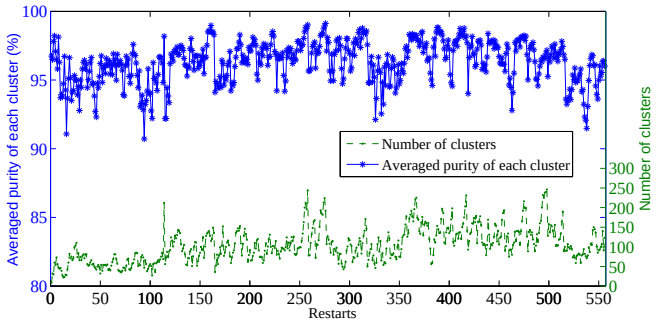


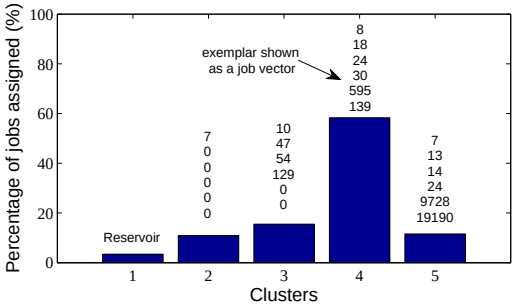
Fig. 6. GSTRAP: Clustering purity (measured at each model rebuilding)

C. Online Monitoring

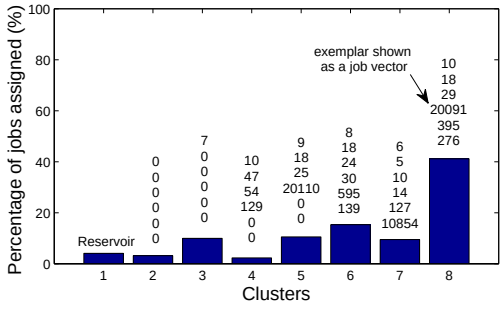
Fig. 7 shows two snapshots yielded by the Online Monitoring module. Each snapshot depicts the current job distribution as a histogram, where each bar corresponds to a cluster (the

corresponding exemplar is given above), the height of which indicates the percentage of the jobs falling in this cluster since the last model rebuilding. For the sake of clarity, only sufficiently representative clusters (including more than 1% of the jobs) are displayed.

The top snapshot (Fig. 7.(a)) is typical of the average job stream distribution. Cluster 4 includes the majority of jobs, which are successfully finished with moderate waiting times (exemplar "[8 18 24 30 595 139]", representing circa 60% of the jobs). Cluster 5 includes other successfully finished jobs; these are computationally demanding (execution time circa 19190s) with a long queuing delay (circa 9728s); this cluster includes about 10% of the jobs. Two types of errors are observed: Cluster 2 (exemplar "[7 0 0 0 0 0]", 10% of the jobs) includes the early stopped jobs (after registration); Cluster 3 (exemplar "[10 47 54 129 0 0]", 15% of the jobs) includes the jobs stopped before arriving at the local computing site. Cluster 1 (Reservoir) indicates the fraction of outliers ($< 3\%$).



(a) Snapshot in a typical day



(b) Snapshot with an anomaly

Fig. 7. GSTRAP: Online Monitoring Day-Views

The bottom snapshot (Fig. 7.(b)) indicates instead an alarming situation. Cluster 8 (exemplar "[10 18 29 20091 395 276]", 40% of jobs) reports an exceptionally long *Ready_for_CE_accept_time* (20,091s, against a few dozen seconds in general), which shows that the LogMonitor (LM) is getting clogged, although the *Ready_for_Transfer_time* remains moderate (29s). As LM feeds the Logging and BookKeeping (L&B), this phenomenon can originate from or result in a temporary failure of the L&B, in fact halting the corresponding WMS. The LM clogging likely also explains Cluster 5 (exemplar "[9 18 25 20110 0 0]"), to be compared with Cluster 4 (exemplar "[10 47 54 129 0 0]", actually very close to Cluster 3 in the top snapshot).

The Online Monitoring module thus yields a compact and understandable summary of the job instant distribution, providing the administrators with a detailed report on the grid activity: the snapshots not only indicate the proportion of good and failed jobs; it also indicates the different time cost of the grid services.

A summary of the snapshots over each restart, day, week or month is provided[6].

D. Offline Monitoring

All exemplars extracted along the online monitoring process can be viewed as jobs that were typical at some moment in the reference period. These exemplars are stored and further clustered using AP to extract *Super Exemplars*; the corresponding clusters likewise are *Super Clusters*.

Let us first consider the erroneous super-exemplars (with label other than *successfully finished*), ordered after the number of services they reached (the first erroneous super-exemplars being "[0 0 0 0 0 0]", no service reached). The frequency and type of errors along the period can be visualized from Fig. 8; the x-axis indicates the day whereas the y-axis is the index of the super-cluster; the gray level of pixel (x, y) indicates the fraction of jobs associated to (an exemplar associated to) super-exemplars y on day x (color is used in the Offline Monitoring for a better visualization; see `http://www.lri.fr/~xlzhang/Grid_monitor/`).

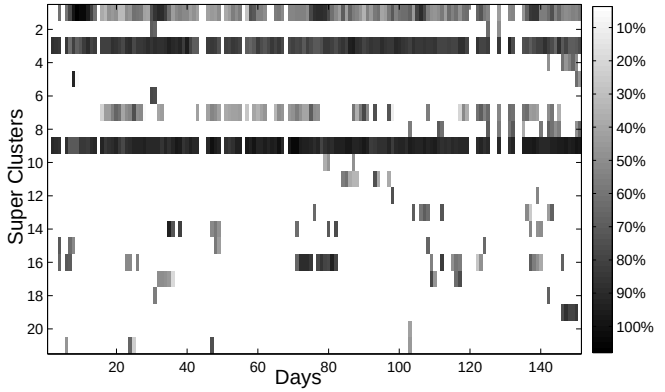


Fig. 8. Offline Monitoring: Frequency of Erroneous Super-Exemplars per day

The most frequent super-exemplars are listed in Table I, where the first column indicates the super-exemplar index in Fig. 8.

After Fig. 8 and Table I, the *Early stopped error* occurred frequently around day10, day32, day78, day106 and day138. The log file actually shows that from 2006-01-07 to 2006-01-13, and from 2006-01-30 to 2006-02-03, a large part of *Early stopped errors* were due to UI_A-1[7]. User Interface

[6]Other monitoring results (avi format) are visible at http://www.lri.fr/~xlzhang/Grid_monitor/.

[7]Identifier omitted for the sake of confidentiality.

TABLE I
OFFLINE MONITORING: MAIN ERRONEOUS SUPER-EXEMPLARS

y	Super exemplars	Main errors
1	0, 0, 0, 0, 0, 0	early stopped
3	145, 0, 0, 0, 0, 0	Cannot plan Unable to receive data
4	7, 0, 226508, 0, 0, 0	Submission to condor failed
5	6, 10, 15, 0, 0, 0	Submission to condor failed transfer failed
7	328, 12, 17, 34, 0, 0	Aborted by user Job proxy is expired
9	109, 98, 116, 164, 0, 0	Aborted by user Job proxy is expired
11	611, 2064, 2069, 2077, 0, 0	Job RetryCount (3) hit
12	410, 10, 20, 3638, 0, 0	cannot retrieve match
14	10, 2893, 4567, 10180, 0, 0	cannot retrieve match
15	15, 120, 298, 15300, 0, 0	cannot retrieve match
16	11, 18, 23, 30602, 0, 0	cannot retrieve match
17	401, 21, 27, 34234, 0, 0	cannot retrieve match
18	5, 33449, 33827, 36859, 0, 0	Job RetryCount (0) hit
19	7, 251275, 271470, 383839, 0, 0	cannot retrieve match
20	8, 66, 592459, 592461, 0, 0	Job RetryCount (7) hit
21	6, 33, 37, 41, 207, 0	Job started to run but lost

UI_B-1 was instead responsible for the majority of the *Early stopped*, and many *Job proxy expired* errors from 2006-03-16 to 2006-03-21. From 2006-05-17 to 2006-05-19, the *early stopped* errors were mostly due to UI_D-1 and UI_A-1 again. After Fig. 8, Super-Exemplar 3 (*Cannot plan*) is among the most frequent types of error; another one, Super-Exemplar 9 mixes different error labels with same time cost of services.

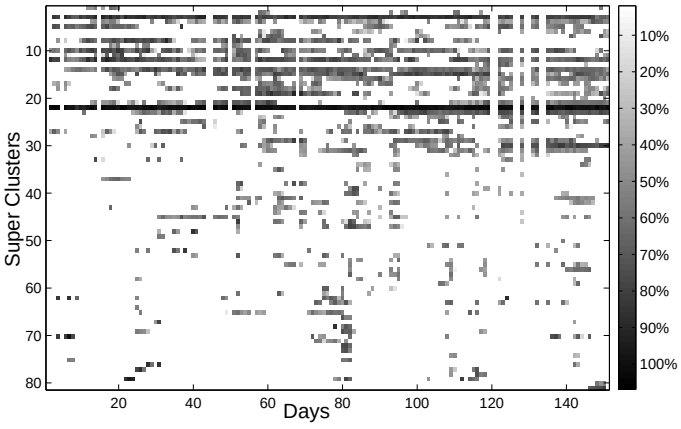


Fig. 9. Offline Monitoring: Frequency of Good Super-Exemplars per day

With regard to the good super-exemplars, let the successfully finished super-exemplars be ordered by increasing *Ready_for_CE_accept_Time*. Fig. 9 displays the representativity of these good super-exemplars in the study period (akin Fig. 8), showing that the most representative ones (Super-Exemplar 1 to 30) are those with smaller *Ready_for_CE_accept_Time*. Some super-exemplars with huge *Ready_for_CE_accept_Time* are visible as light spots at the left bottom of Fig. 9. Super-Exemplar 77 ("[6 11 17 47938 40 49]") and 79 ("[5 10 15 195017 56 185]") are represented mostly on days 23 to 27. The log files indeed show that the jobs submitted by UI_A-1 to RB_A-2 from Jan. 22nd to 27th have very high *Ready_for_CE_accept_Time*.

The purity of the super clusters associated to these good super-exemplars (likewise ordered by increasing *Ready_for_CE_accept_Time*) is displayed on Fig. 10, together with the overall size of the super-cluster (in log scale). Notably, these super-cluster purity is circa 95% (much higher than the average purity along time, Fig. 6) except for the last three clusters. The latter super-exemplars have a huge *Ready_for_CE_accept* (ranging from 21 hours to 100 hours); although successfully finished, these jobs are indeed similar to quite a few bad jobs.

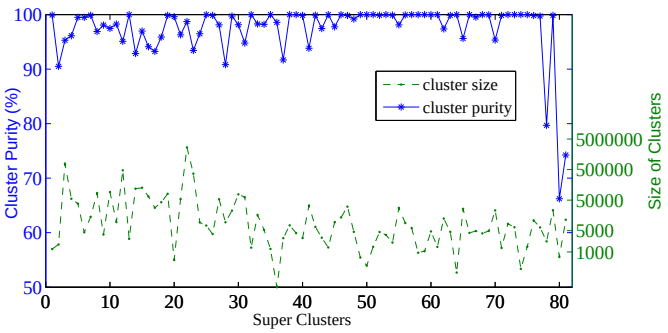


Fig. 10. Offline Monitoring: Purity of Super Clusters

VI. CONCLUSION

The GSTRAP system presented in this paper is a real-time Grid Monitoring system, providing the administrator with online and offline views of the stream of jobs submitted to the EGEE grid. The main contribution of this work is to actually provide an intuitive and accurate view of the EGEE load, along several dimensions. Firstly, the current distribution of the jobs is displayed as a histogram, amenable to immediate interpretation and detection of alarms (Fig. 7). Secondly, the sequence of the model rebuilding gives an intuitive idea of the load dynamics. Lastly, the Offline Monitoring module offers a consolidated view of the load, enabling to inspect its long-term trends *a posteriori*. Overall, these functionalities can be viewed as a first step toward extracting manageable, understandable and valuable summaries from the gLite traces [4].

This system suffers from several limitations. In the short term, the summary quality can be enhanced by increasing the cluster accuracy, e.g. by using an educated distance among jobs; distance metric currently is among the hottest topics in Supervised Machine Learning [31]. Independently, the λ parameter actually governs the sensitivity of the STRAP algorithm and the number of model rebuilding; self-adapting the sensitivity of the change detection test is a perspective for further research. In the longer-term, while GSTRAP current processing rate is circa 40,000 jobs per minute, a better scalability will be needed to face the increase of the computational load.

Further extensions of GSTRAP are envisioned. A most straightforward step will be to consider a more comprehen-

sive description of the jobs, e.g., related to User Interface and Computing Elements. In collaboration with the system administrators and the operation teams, anomalous situations will be listed and related to "alarming" exemplars, aiming at the prevention of anomalies. Independently, the proposed approach will be extended to aggregate and describe the computational load at the user level (as opposed to, at the job level), and at the virtual organization level, in order to provide customized and more user-friendly services.

ACKNOWLEDGMENT

This work has been partially supported by the European Infrastructure Project EGEE-III INFSO-RI-222667 and the Network of Excellence PASCAL, IST-2002-506778.

REFERENCES

[1] I. Rish, M. Brodie, S. Ma, and et al, "Adaptive diagnosis in distributed systems," *IEEE Transactions on Neural Networks*, vol. 16, pp. 1088–1109, 2005.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, 2003.

[3] E. Laure, S. Fisher, A. Frohner, C. Grandi, and et al., "Programming the grid with gLite," *Computational Methods in Science and Technology*, vol. 12, pp. 33–45, 2006.

[4] R. Jones, "The EGEE project," Talk at EGEE'08 Conference, 2008, http://indico.cern.ch/contributionDisplay.py?contribId=116&sessionId=0&confId=32220.

[5] X. Zhang, C. Furtlehner, and M. Sebag, "Data streaming with affinity propagation," in *European Conference on Machine Learning and Practice of Knowledge Discovery in Databases, ECML/PKDD*, 2008, pp. 628–643.

[6] E. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, pp. 100–115, 1954.

[7] D. Hinkley, "Inference about the change-point from cumulative sum tests," *Biometrika*, vol. 58, pp. 509–523, 1971.

[8] R. Byrom, D. Colling, S. M. Fisher, and et al., "Performance of R-GMA based grid job monitoring system for CMS data production," in *IEEE Nuclear Science Symposium Conference Record*, 2005, pp. 860–864.

[9] E. Imamagic and D. Dobrenic, "Grid infrastructure monitoring system based on Nagios," in *GMW '07: Proceedings of the 2007 workshop on Grid monitoring*, 2007, pp. 23–28.

[10] C. C. Cirstoiu, C. C. Grigoras, L. L. Betev, and et al., "Monitoring, accounting and automated decision support for the alice experiment based on the MonALISA framework," in *GMW '07: Proceedings of the 2007 workshop on Grid monitoring*, 2007, pp. 39–44.

[11] G. Donvito, G. Tortone, M. Maggi, and et al., "Job-monitoring over the grid with GridIce infrastructure," CERN, Tech. Rep. EGEE-PUB-2004-008, 2005.

[12] H. linh Truong and T. Fahringer, "SCALEA-G: a unified monitoring and performance analysis system for the grid," *Scientific Programming*, vol. 12, pp. 225–237, 2004.

[13] A. Křenek, J. Sitera, L. Matyska, and et al., "gLite job provenance—a job-centric view," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 453–462, 2008.

[14] R. Kalmady, D. Sonvane, P. Chand, and et al., "Monitoring the availability of grid services using SAM and gridview," in *International Symposium on Grid Computing ISGC 2007*, ser. LLC, 2007, pp. 163–168.

[15] "LEMON - LHC Era Monitoring." [Online]. Available: http://lemon.web.cern.ch/lemon/docs.shtml

[16] N. Palatin, A. Leizarowitz, A. Schuster, and et al., "Mining for misconfigured machines in grid systems," in *International conference on Knowledge discovery and data mining(KDD)*, 2006, pp. 687–692.

[17] J. Andreeva, B. Gaidioz, J. Herrala, and et al., "Dashboard for the LHC experiments," *Journal of Physics: Conference Series*, vol. 119, 2008.

[18] Real Time Monitor: http://gridportal.hep.ph.ic.ac.uk/rtm/.

[19] B. Kryza, R. Slota, M. Majewska, and et al., "Grid organizational memory-provision of a high-level grid abstraction layer supported by ontology alignment," *Future Gener. Comput. Syst.*, vol. 23, pp. 348–358, 2007.

[20] W. Xing, M. D. Dikaiakos, and R. Sakellariou, "A core grid ontology for the semantic grid," in *IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, 2006, pp. 178–184.

[21] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, pp. 41–49, 2003.

[22] A. Mutz, R. Wolski, and J. Brevik, "Eliciting honest value information in a batch-queue environment," in *IEEE/ACM International Conference on Grid Computing*, 2007, pp. 291–297.

[23] P. A. Dinda, "Online prediction of the running time of tasks," *Cluster Computing*, vol. 5, pp. 225–236, 2002.

[24] T. Fahringer, R. Prodan, R. Duan, and et al., "ASKALON: A grid application development and computing environment," in *IEEE/ACM International Workshop on Grid Computing*, 2005, pp. 122–131.

[25] H.-L. Truong, T. Fahringer, and S. Dustdar, "Dynamic instrumentation, performance monitoring and analysis of grid scientific workflows," *Journal of Grid Computing*, pp. 1–18, 2005.

[26] C. Germain-Renaud and D. Monnier-Ragaigne, "Grid result checking," in *CF '05: Proceedings of the 2nd conference on Computing frontiers*, 2005, pp. 87–96.

[27] L. F. G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," *Future Gener. Comput. Syst.*, vol. 18, pp. 561–572, 2002.

[28] G. Cormode, S. Muthukrishnan, and W. Zhuang, "Conquering the divide: Continuous clustering of distributed data streams," in *ICDE*, 2007, pp. 1036–1045.

[29] J. Gama and P. P. Rodrigues, "Stream-based electricity load forecast," in *European conference on Principles and Practice of Knowledge Discovery in Databases(PKDD)*, 2007, pp. 446–453.

[30] B. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[31] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2005.