An Approach to Model Resources Rationalisation in Hybrid Clouds through Users Activity Characterisation

Giovanni Battista Barone, Davide Bottalico, Rosanna Campagna and Giuliano Laccetti
University of Naples Federico II
Naples, Italy
Emails: {gbbarone, davide.bottalico, rosanna.campagna, giuliano.laccetti}@unina.it

Vania Boccia
Italian National
Institute of Nuclear Physics
Seat of Naples, Italy
Email: vania.boccia@na.infn.it

Luisa Carracciuolo
Italian National
Research Council
Naples, Italy
Email: luisa.carracciuolo@cnr.it

***Abstract*—In recent years, some strategies (e.g., server consolidation by means of virtualisation techniques) helped the managers of large Information Technology (IT) infrastructures to limit, when possible, the use of hardware resources in order to provide reliable services and to reduce the Total Cost of Ownership (TCO) of such infrastructures. Moreover, with the advent of Cloud computing, a resource usage rationalisation can be pursued also for the users applications, if this is compatible with the Quality of Service (QoS) which must be guaranteed. In this perspective, modern datacenters are "elastic", i.e., able to shrink or enlarge the number of local physical or virtual resources from private/public Clouds. Moreover, many of large computing environments are integrated in distributed computing environment as the grid and cloud infrastructures. In this document, we report some advances in the realisation of a utility, we named *Adaptive Scheduling Controller (ASC)* which, interacting with the datacenter resource manager, allows an effective and efficient usage of resources, also by means of users jobs classification. Here, we focus both on some data mining algorithms which allows to classify the users activity and on the mathematical formalisation of the functional used by ASC to find the most suitable configuration for the datacenter's resource manager. The presented case study concerns the SCoPE infrastructure, which has a twofold role: local computing resources provider for the University of Naples Federico II and remote resources provider for both the Italian Grid Infrastructure (IGI) and the European Grid Infrastructure (EGI) Federated Cloud.**

Keywords–Adaptive scheduling and resources management; Virtualisation and Cloud computing; large scale and distributed systems; data analysis.

I. INTRODUCTION

The TCO of large scale computing systems, with an emphasis on systems built to support a wide range of users and different applications includes, among others, the initial hardware cost (for computing nodes, storage systems, racks, facilities, etc.), the personnel/system administrator costs (salaries for software and hardware maintenance requiring specialised know-how), business premises, and energy costs (which includes the additional power requirements for cooling and power delivery inefficiencies) [1].

The aim of a good system manager is the TCO minimisation, that can be performed also by means of the resources rationalisation (e.g., through the reduction of the number of active resources on the basis of the effective jobs request, the use of virtual resources by means of cloud computing paradigms, and so on). All said, however, without neglecting users satisfaction. Users of a large general purpose system can have conflicting demands (i.e., short versus long jobs, sequential versus parallel applications, etc.). For this reason the achievement of all user satisfaction is a hard task for any system manager.

In our previous works (e.g., in [2]), we described some issues related to the design and the implementation of a control system (ASC) which, using an *"adaptive"* approach in job scheduling policy, allows a balanced, effective and efficient use of computational resources. In particular, ASC, on the basis of a *jobs classification*, is asked to dynamically identify, given historical data, the most *"suitable"* configuration of the resource manager/scheduling systems for the current jobs work flow.

With the advent of new recent distributed paradigms, e.g., Cloud computing, modern datacenters are being more "elastic" (according to the U.S. National Institute of Standards and Technology (NIST) characterisation of Cloud computing based infrastructures) [3]), i.e., able to shrink or enlarge on demand the number of active resources, by recruiting both local physical (and virtual) resources and remote resources from public Clouds. Moreover, many of large computing environments are integrated in distributed computing environment as the grid and cloud infrastructures. All these above described issues increase the dynamic nature of the current computing environments.

In this paper, we provide a description of the advancements in the mathematical formalisation and implementation of ASC for modern datacenters and we report the work made toward a complete jobs classification by means of data mining techniques.

In Section II, we provide some details on the related works in the field of adaptive scheduling, focusing on all the key differences between the approaches existing in literature and ours; in Section III, we report a short description of the ASC system, with a focus on its mathematical formalisation; in Section IV, we describe the case study of the SCoPE computing infrastructure at University of Naples Federico II reporting some results related to the use of *Data Mining* techniques for job classification in a "production context".

 ISBN: 978-1-61208-389-6

The SCoPE infrastructure has a twofold role: local computing resources provider for the University of Naples Federico II and remote resources provider for both the IGI [4] and the EGI Federated Cloud [5].

II. Related work

An adaptive solution to the scheduling problem, in the sense used by Casavant and Kuhl [6], is able to change dynamically algorithms and parameters that define the scheduling policy, according to the previous and current behaviour of the system in response to previous decisions taken by the scheduling system itself.

The most common property to search for a scheduler (or resource management subsystem) is dynamicity. In a dynamic scenario, the scheduler takes into account the current state of affairs as it perceives them in the system. This is done during the normal operation of the system under a dynamic and unpredictable load. In an adaptive system, the scheduling policy itself reflects changes in its environment (the running system).

A preliminary approach in jobs scheduling, such as those described by Serazzi and Calzarossa [7], exists in modelling adaptive control systems able to maximise a given performance criterion, such as system throughput. However, in the last years the heterogeneity of applications using general purpose computational grows together with the complexity of resource requirements. Maximising system throughput shouldn't longer be the only requirement for a scheduling scheme. The quality of service perceived by the user offers an instructive example where the solution of the problem is achieved in a *"acceptable"* time [8]. Recent approaches in jobs scheduling take into account both *efficiency* and *fairness* for homogeneous workloads [9], but the open challenge is to achieve the same goal for *not* homogeneous workloads.

If we have in mind what Feitelson says in [10]:

> *In reality, the metrics attempt to formalise the real goal of a scheduler:*
> - *Satisfy the users,*
> - *Maximise the profit.*

more sophisticated and finer-grained resource coordination mechanisms are required.

If we consider the problem of user communities heterogeneity and dynamicity using a large scale computational resource, an *"adaptive"* approach to the job scheduling seems to be a promising way for getting the right trade-off among different, and often conflicting, classes of applications demands sharing the same set of resources. Moreover, the overall set of resources, generally included in a modern "elastic" datacenter, is dynamic and heterogeneous itself (because of the chance to include different resources from Clouds).

Here, we propose a pragmatic approach to solve this kind of problem. The approach is led by the fact that we consider the overall set of users organised in communities each of them having almost homogeneous requirements. The above conditions induce a classification of the users (or equivalently of the jobs) into different classes: the terms "users" and "jobs" are intended to represent the same entities.

The idea to organise jobs into groups to improve the solution of scheduling problem on complex computing systems, from High-Performance Computing (HPC) system to the distributed infrastructures, is not new [11][12][13]. The principal aim of all those works is to present and to discuss the task distribution problem onto HPC and distributed systems to improve performance and load balancing of the applications.

Our approach is based, instead, on the idea of organising jobs in groups or *clusters "naturally"* deriving from the nature of our audience. The aims are mainly 1) to achieve the efficiency of the entire system rather than the performance of a single application and 2) to reduce the computational complexity for the optimisation problem solution required to find the most *"suitable"* configuration of the system.

To get an automatic classification of jobs we use Data Mining procedures, i.e., *"data clustering"*. The term *"data clustering"* refers to the process of the Data Mining techniques aimed to divide into natural groups the instances represented by data [14].

There are different ways to represent results of clustering. The groups to be identified may be exclusive (so that any instance belongs to only one group) or may be overlapping (so that an instance may fall into several groups). Besides, they may be probabilistic, whereby an instance belongs to each group with a certain probability. They may be hierarchical, such that there would be a crude division of instances into groups at the top level, and each of these groups is refined further-perhaps all the way down to individual instances. In this work, we consider algorithms building clusters in numerical domains, partitioning instances into disjoint clusters.

According to the approach described by Estivill-Castro [15], in order to define the *"data clustering"* problem we have to:

- express assumptions on the model describing the nature of the data (e.g., a probabilistic model),
- formulate the mathematical model for the problem (e.g., an optimisation problem)
- choose the solving algorithm for the problem (e.g., the *k-means* algorithm [14])

So, as starting point and *"proof of concept"* for the validation of our approach, we considered algorithms based on the classic clustering algorithm *k-means*. Such algorithm can usefully be used to iteratively compute an approximation to the solution of the minimisation problem of a functional, based on an Euclidean norm, depending on data with multivariate normal distribution [15].

In the next section, we provide a mathematical formalisation for ASC based on the hypothesis that a heterogeneous work flow can be partitioned into homogeneous classes of jobs. The way we used to do this partitioning is described in Section IV.

III. ASC: architecture and operating model

A. Architecture description

We call "adaptive" a system able to "reconfigure" itself on the basis of changing in user typology. Such a mechanism, analysing system behaviour by some *key-statistics* (e.g depending on queue waiting time, jobs throughput, resource usage, and so on), dynamically defines a new set of scheduler *key-parameters* values. The scheduler's new configuration has to meet both user satisfaction and efficiency/productivity in computational resource usage.

 ISBN: 978-1-61208-389-6

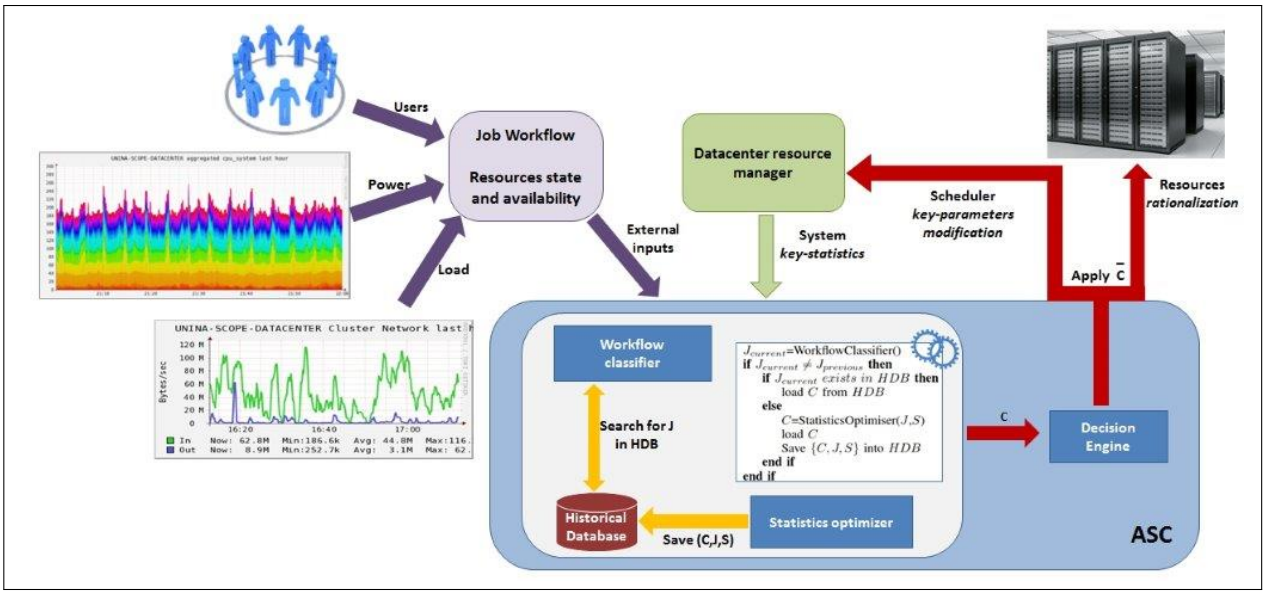

Figure 1. An *"adaptive"* approach in job scheduling policy by an external controller

From now on, $\mathbf{C}$ is the set of the resource manager configurations (each of them is identified by a set of value for *key-parameters*), $\mathbf{J}$ is the vector representing the work flow (partitioned into m classes of homogeneous jobs), and $\mathbf{S}$ contains the values of the considered metrics.

Looking at Figure 1, the ASC engine is organised into four core software modules, each one with a specific role in the operative model:

- the *Work flow Classifier* is asked to perform a job classification into m classes of homogeneous jobs starting from the set of n heterogeneous jobs;
- the *Historical Database* has the role to save for each meaningful classified work flow J, the set C of values for *key-parameters*, obtained from the *Statistics Optimiser* that optimise the considered *key-statistics* S;
- the *Statistics Optimiser* provides the new set of values for C;
- the *Decision engine*, implements a new asset (the $\overline{C}$ configuration) for the datacenter by choosing physical or virtual resources from local or public Clouds also on the basis of other criteria (e.g., resource power consumption and geographical location).

The algorithm in Figure 2 describes the interaction among ASC core modules; particularly, $J_{current}$ refers to the new work flow, while $J_{previous}$ is the last classified work flow; HDB refers to the Historical Database.

B. Statistics Optimiser modelling

In our first ASC formalisation (see [2]), we obtained:

$$\mathbf{C} = \sum_{j=1}^{m} \alpha_j(\mathbf{J}) \cdot F_j(\mathbf{S}) \qquad (1)$$

```
1:  ...
2:  loop
3:      J_current=WorkflowClassifier()
4:      if J_current ≠ J_previous then
5:          if J_current exists in HDB then
6:              load C from HDB
7:          else
8:              C=StatisticsOptimiser(J,S)
9:              load C
10:             Save {C, J, S} into HDB
11:         end if
12:     end if
13:     C̄=DecisionEngine(C)
14:     apply(C̄)
15:     Sleep (some time)
16: end loop
17: ...
```

Figure 2. ASC engine algorithm

The symbol $(\cdot)$ denotes a suitable operator, each function F_j computes *optimal* parameters values for job type j while α_j expresses the *weight* to be considered for job type j.

For each job class $j = 1, ..., m$, we consider the following classic *key-statistics* (the first is representative of the whole computing system efficiency, while the second and third ones express user satisfaction):

System effectiveness ratio $E^{(j)} = \frac{\sum_{i=1}^{n} p_i t_i}{PT}$;

System Make span $M_k^{(j)} = \max_{i=1,...,n} (t_i + q_i)$;

Queue waiting time average $Q^{(j)} = \frac{\sum_{i=1}^{n} q_i}{n}$

where P is the total number of available processors, n is the total number of jobs for the work flow, Q is the total queue time for all the jobs of the work flow, T is the wall clock run time for all jobs completion, p_i, t_i and q_i are respectively the

 ISBN: 978-1-61208-389-6

number of processors requested, the execution time and the queue time for the job i.

Since:

$$M_k^{(j)} = \|\underline{t} + \underline{q}\|_\infty \geq \left| \|\underline{t}\|_\infty - \|\underline{q}\|_\infty \right|$$

and because of the equivalence of norms, we have:

$$M_k^{(j)} \quad \geq \quad \left| \|\underline{t}\|_1 - \|\underline{q}\|_1 \right| = \left| \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} q_i \right|$$

Under the *realistic* assumption:

$$\sum_{i=1}^{n} q_i < \sum_{i=1}^{n} t_i \tag{2}$$

we have:

$$\sum_{i=1}^{n} t_i \leq M_k^{(j)} + nQ^{(j)}$$

so, for each job class $j = 1, ..., m$, we want to solve the following:

Problem: *To compute the set of the scheduler* key-parameters $C_{Opt}^{(j)}$ *such that*

$$\mathbf{C}_{Opt}^{(j)} = F_j \left(\mathbf{S}_{Opt}^{(j)} \right) \tag{3}$$

where $\mathbf{S}_{Opt}^{(j)} = \left(E_{Opt}^{(j)}, M_k{}_{Opt}^{(j)}, Q_{Opt}^{(j)} \right)$ *and* $E_{Opt}^{(j)}, M_k{}_{Opt}^{(j)}, Q_{Opt}^{(j)}$ *are the solutions of the constrained "optimisation" problem:*

$$\begin{cases} \max\{E^{(j)}\} \qquad s.t. \\ T = \sum_{i=1}^{n} t_i \leq (M_k{}^{(j)} + nQ^{(j)}) \\ \sum_{i=1}^{n} q_i < \sum_{i=1}^{n} t_i \end{cases} \tag{4}$$

$\Diamond$

We highlight that the problem (4) includes also the two limit cases:

best case: $q_i = 0 \; \forall i$, i.e., $Q^{(j)} = 0$ (there are no jobs in system queues)

$$\begin{aligned} M_k^{(j)} &= \max_{i=1,...,n} (t_i + q_i) = \max_{i=1,...,n} t_i \\ E^{(j)} &= \frac{\sum_{i=1}^{n} t_i p_i}{P \max_{i=1,...,n} t_i} \leq \frac{\sum_{i=1}^{n} t_{max} p_i}{P t_{max}} \approx 1 \end{aligned} \tag{5}$$

In the ideal case, there aren't new jobs in queue and all the processors in the system are allocated for jobs:

$$P = \sum_{i=1}^{n} p_i \tag{6}$$

so the system reaches the maximum efficiency ($E = 1$).

worst case: $q_i \to \infty \; \forall i$. This case is not possible because of the (2). Thus, being at most: $q_i \to t_i \; \forall i$ we have:

$$\begin{aligned} M_k^{(j)} &= \max_{i=1,...,n} (t_i + q_i) \to \max_{i=1,...,n} (2t_i) \\ E^{(j)} &\to \frac{\sum_{i=1}^{n} t_i p_i}{P \max_{i=1,...,n} (2t_i)} \leq \frac{\sum_{i=1}^{n} t_{max} p_i}{P 2 t_{max}} \approx \frac{1}{2} \end{aligned} \tag{7}$$

so even if the (6) is verified, system efficiency E results under the 50%.

Both in (5) and in (7), t_{max} is the maximum wall clock time allowed on the queues.

The scheduler *key-parameters* value affects the queue time for different job types and therefore the values of the metrics considered above. On the basis of some information provided from the system:

- the fixed maximum execution time t_{max} for each job i submitted to a certain queue of the scheduling system;
- the total amount of requested processors P;
- the user request of processors for each job i

we can estimate the metrics E, Q and M_k only if a probability distribution function for the queue time values q_i is known for all the jobs in the work flow.

Presently, we are working to define a stochastic model to forecast queue times also thanks to the already classified work flows.

IV. A CASE STUDY FOR THE ASC WORK FLOW CLASSIFIER

In this section, we report some experiences related to the use of data mining techniques to automate the work flow partitioning into homogeneous job classes as described in Section III. We use computational resources at the University of Naples Federico II, acquired in the context of PON *Sistema Cooperativo Per Elaborazioni scientifiche multidisciplinari* (S.Co.P.E.) Italian National project [16].

SCoPE resources are made available to national and international relevant distributed infrastructures (IGI and EGI) and, thus, used not only by the local users.

Due to the user communities heterogeneity, computational resources are used both for *"traditional"* GRID jobs and for HPC applications. From a heuristic analysis, we observe that SCoPE jobs are applications mostly sequential or with a low degree of parallelism (DOP) with a short duration. Just a subset, however large enough of SCoPE jobs, has a medium-high DOP and a more long duration.

The computational resources (about 2000 cores) are accessed by submitting jobs to the Resource Management System (based on Maui-Torque systems).

To automate the definition of the jobs classification **J**, we consider a set of clustering algorithms implemented by the *Waikato Environment for Knowledge Analysis (WEKA)* package [17]. WEKA is a well-known suite of machine learning software which supports several typical data mining processes, particularly data preprocessing, clustering, classification, regression, visualisation, and feature selection.

ARFF format is used by WEKA to represent data sets that consist of independent, unordered instances. Each instance is characterised by its values on a fixed, predefined set of features or attributes.

Since we are interested on a jobs classification based on the duration and on the DOP of each job, we chose to represent each job with the two following attributes:

`ntasks` the number of concurrent tasks of the job

`tte` the job Total Time of Execution

The clustering processes are performed, by the three following algorithms and related WEKA command lines, on data related to the last 2 years (over 3 millions jobs):

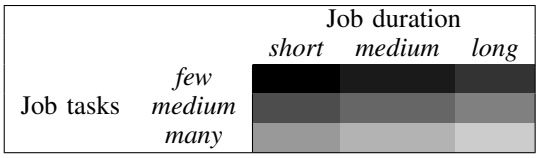

Figure 3. Jobs *"ideal classification"*

SKMeans: provides clustering with the k-means algorithm [14].

```
weka.clusterers.SimpleKMeans  -N 9
                -A "weka.core.EuclideanDistance
                -R first-last" -I 500 -S 10
```

XMean: provides k-means extended by an *"Improve-Structure part"* and automatically determines the number of clusters [18].

```
weka.clusterers.XMeans -I 1 -M 1000 -J 1000
                -L 2  -H 9 -B 1.0 -C 0.5
                -D "weka.core.EuclideanDistance
                -R first-last" -S 10
```

Festivity: provides the *"farthest first traversal algorithm"*, which works as a fast simple approximate *"clusterer"* modeled after simple k-means [19].

```
weka.clusterers.FarthestFirst  -N 9 -S 1
```

For all the algorithms, we chose that the maximum possible number of clusters is $Ncluster_{max} = 9$ (see the red highlighted parameters in the command lines above) on the basis of the *"ideal classification"* represented in Figure 3.

The aim of the tests described above was to identify the most effective clustering algorithm in terms of:

1) computational cost
2) compliance to the results of our heuristic jobs classification

All the algorithms build at least four classes, where the most numerous one collects sequential jobs (or with low DOP jobs) consistently with our heuristic classification.

Anyway, the *SKMeans* based algorithm seems to be the most useful choice because, from our point of view, it responds better to the point 2 above: it builds more different classes, each of them with a significative number of elements (see Table I). The *SKMeans* based algorithm is, in terms of computational cost, more expensive than other ones but its cost can be considered acceptable compared to the frequency of scheduler reconfiguration (taking into account the mean duration of the jobs, we can assume a period of few days).

Figures 4 and 5 show, respectively, the real work flow characterisation and the results of the clustering process performed by WEKA on data related to the work flow execute on SCoPE resources during the month of March 2014. The clustering processes are performed by the *SKMeans* based algorithm which we choose as the *"optimal"* one for our purpose. During the considered month, different type of work flows and jobs are present on SCoPE infrastructure. A similar behaviour is

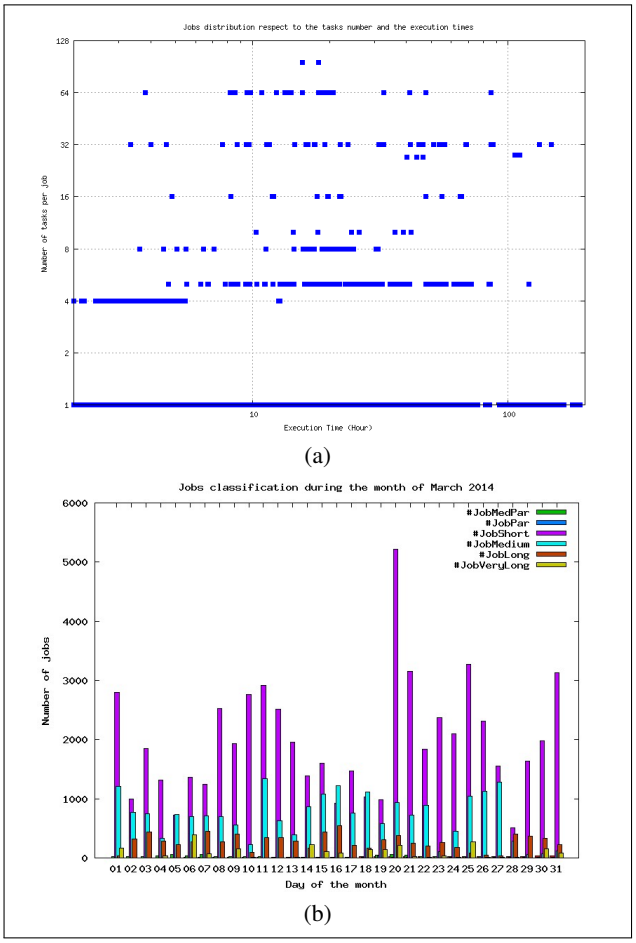

Figure 4. March 2014: real job work flow representation by two different views: the jobs distribution both respect to the number of tasks and to the execution time (a); the trend of the jobs number as a function of the month days (b).

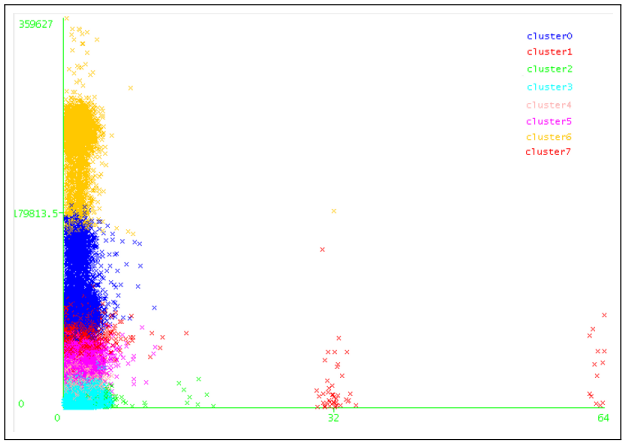

Figure 5. March 2014: clustering processes results on SCoPE jobs

TABLE I. CLUSTERING PROCESSES RESULTS ON ALL SCOPE JOBS

SKMeans (Execution times: 786.33 secs)

Clusters	# of Elements	% of Elements
0	5804	0
1	11117	0
2	265923	7
3	52620	1
4	337429	9
5	417363	11
6	83707	2
7	2472225	68

XMeans (Execution times: 87.16 secs)

Clusters	# of Elements	% of Elements
0	11333	0
1	142700	4
2	10937	0
3	3481218	95

FarthestFirst (Execution times: 12.75 secs)

Clusters	# of Elements	% of Elements
0	3639639	100
1	3	0
2	30	0
3	106	0
4	192	0
5	97	0
6	12	0
7	4916	0
8	1193	0

present on other months of the year, confirming the need for adaptive approach to the scheduling problem.

V. CONCLUSION AND FUTURE WORK

In this document, we described the progresses made to devise ASC, which aims to gain a balanced, efficient and effective use of computing resources by heterogeneous users communities. Here, we gave details about ASC mathematical formalisation focusing on the chance to have a computable estimation for the involved *key-statistics*; moreover the use of data mining techniques allowed us to build a job clustering into homogeneous classes, starting from real heterogeneous jobs work flows.

Presently, we are working on:

- the dynamic work flow classification starting from job clustering results here obtained;
- the mathematical model with the aim to define, for different sets of *key-parameters* and already classified work flows, the probability distribution function for queue times.
- the realisation of a historical database of scheduler configurations, known work flows and related *key-statistics* values;
- the deployment of some features to enable ASC to take into account other parameters, e.g., computing nodes availability/dependability and metrics related to environmentally conscious computing services [20].

ACKNOWLEDGEMENT

This work is part of the activities of a multidisciplinary group (GTT), responsible for the SCoPE infrastructure management. It has been realised thanks to the use of the SCoPE computing infrastructure at the University of Naples, also in the framework of PON *"Rete di Calcolo per Superb e le altre applicazioni"* (ReCaS) project.

REFERENCES

[1] A. Nazir and S.-A. Srensen, "Cost-benefit analysis of high performance computing infrastructures," in SOCA. IEEE, 2010, pp. 1–8.

[2] G. Barone, V. Boccia, D. Bottalico, L. Carracciuolo, A. Doria, and G. Laccetti, "Modelling the behaviour of an adaptive scheduling controller," in Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on, July 2012, pp. 438–442.

[3] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [retrieved: February, 2015]

[4] Italian Grid Infrastructure. [Online]. Available: http://www.italiangrid.it/ [retrieved: February, 2015]

[5] Federated cloud, Building a grid of clouds. [Online]. Available: https://www.egi.eu/infrastructure/cloud/ [retrieved: February, 2015]

[6] T. Casavant and J. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," IEEE Transactions on Software Engineering, vol. 14, no. 2, 1988, pp. 141–154.

[7] G. Serazzi and M. Calzarossa, "Adaptive Optimization of a System's Load," IEEE Trans. Software Eng., vol. 10, no. 6, 1984, pp. 837–845.

[8] I. Foster. What's faster a supercomputer or EC2? [Online]. Available: http://ianfoster.typepad.com/blog/2009/08/whats-fastera-supercomputer-or-ec2.html [retrieved: February, 2015]

[9] H. Sun, Y. Cao, and W.-J. Hsu, "Fair and Efficient Online Adaptive Scheduling for Multiple Sets of Parallel Applications," in ICPADS. IEEE, 2011, pp. 64–71.

[10] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K. Sevcik, and P. Wong, "Theory and practice in parallel job scheduling," in Job Scheduling Strategies for Parallel Processing, ser. Lecture Notes in Computer Science, D. Feitelson and L. Rudolph, Eds. Springer Berlin Heidelberg, 1997, vol. 1291, pp. 1–34.

[11] K. Gkoutioudi and H. D. Karatza, "Task cluster scheduling in a grid system," Simulation Modelling Practice and Theory, vol. 18, no. 9, 2010, pp. 1242–1252.

[12] J. Aguilar and E. Gelenbe, "Task Assignment and Transaction Clustering Heuristics for Distributed Systems," Information Sciences, vol. 97, 1997, pp. 1–2.

[13] Z. C. Papazachos and H. D. Karatza, "The impact of task service time variability on gang scheduling performance in a two-cluster system." Simulation Modelling Practice and Theory, vol. 17, 2009, pp. 1276–1289.

[14] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[15] V. Estivill-Castro, "Why so many clustering algorithms: A position paper," SIGKDD Explor. Newsl., vol. 4, no. 1, Jun. 2002, pp. 65–75.

[16] Merola L. on behalf of S.Co.P.E. Project, "The S.Co.P.E. Project," in Proceedings of the Final Workshop of the Grid Projects of the Italian National Operational Programme 2000-2006 Call 1575, Edited by Consorzio COMETA, 2008, pp. 18–35.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," SIGKDD Explor. Newsl., vol. 11, no. 1, Nov. 2009, pp. 10–18.

[18] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in Proceedings of the Seventeenth International Conference on Machine Learning, ser. ICML '00. Morgan Kaufmann Publishers Inc., 2000, pp. 727–734.

[19] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," Mathematics of Operations Research, vol. 10, 1985, pp. 180–184.

[20] G. Barone, R. Bifulco, V. Boccia, D. Bottalico, and L. Carracciuolo, "Toward a flexible, environmentally conscious, on demand high performance computing service," in Data Compression, Communications and Processing (CCP), 2011 First International Conference on, June 2011, pp. 136–138.

 ISBN: 978-1-61208-389-6