Autonomic Management of Workflows on Hybrid Grid-Cloud Infrastructure

Giuseppe Papuzzo and Giandomenico Spezzano
CNR – National Research Council of Italy
Institute for High Performance Computing and Networking (ICAR)
Via P. Bucci 41C - 87036 Rende (CS), Italy
{papuzzo, spezzano}@icar.cnr.it

***Abstract*—Grids can be considered as dominant platforms for large-scale parallel/distributed computing in science and engineering. Clouds allow users to acquire and release resources on-demand. Next generation computing environments will benefit from the combination of Grid and Cloud paradigms providing frameworks that integrate traditional Grid services with on-demand Cloud services. Nowadays, workflows are the preferred means for the combination of services into added value service chains representing functional business processes or complex scientific experiments. A promising way to manage effectively services composition in a dynamic and heterogenous environment is to make the workflow management framework able to self-adapt at runtime to changes in its environment and provide an uniform resource access mechanism over Grid and Cloud infrastructures. Autonomic workflow management systems can support the runtime modification of workflows with the aim of improving their performance and recover from faults determining and provisioning the appropriate mix of Grid/Cloud services with requested QoS.**

This paper describes Sunflower an innovative P2P agent-based framework for configuring, enacting, managing and adapting workflows on hybrid Grid-Cloud infrastructures. To orchestrate Grid and Cloud services, Sunflower uses a bio-inspired autonomic choreography model and integrates the scheduling algorithm with a provisioning component that can dynamically launch virtual machines in a Cloud infrastructure to provide on-demand services in peak-load situations.

INTRODUCTION

Workflow management systems (WMSs) are generally utilized to define, manage and execute workflow applications on Grid resources [1]. WMSs must be dynamic and adaptive as Web/Grid services may be subjected to high variability in demand and suffer from unpredictable peaks of heavy load and/or the resources availability can change over time while the workflow is executing. When there are insufficient resources to meet the QoS requirement of the application, the system should provide additional computational resources. In these cases, the ability of temporarily extending the capacity of the Grid by renting resources from an external provider can be a viable proposition. Such an opportunity is offered by Cloud Computing [2] which, by leveraging virtual machine technology, delivery IT infrastructure on demand on *a pay per use* basis. Clouds allow users to acquire and release services on-demand. These services can be Infrastructure as a Service (*IaaS*), Platform as a Service (*PaaS*) and Software as a Service (*SaaS*).

Next generation computing environments will benefit from the combination of Grid and Cloud paradigms providing frameworks that integrate traditional Grid services with on-demand Cloud services. This enable grid workflow systems to easily grow and shrink the available resource pool as the needs of the workflow change over time. A workflow running on a hybrid computing infrastructure should react to workload variations by altering its configuration in order to optimally use the Grids and Clouds available resources and recover from faults. Such modifications should happen automatically and without any human intervention. An autonomic WMS [3] exhibits the ability to reconfigure itself to the changes in the environment and can adapt the size to keep the balance between servicing its workload with optimal performance and ensuring efficient resources allocation.

In this paper, we present a Cloud extension of Sunflower [4] a bio-inspired service-based framework for the execution of autonomic workflows that orchestrate Grid and Cloud services using a decentralized choreography model. Sunflower normally runs workflows using the grid resources but when the performance of web services degrades and the QoS no longer meets the needs of a particular virtual organization or a particular member of a virtual organization, it acquires news hosts using a Cloud computing infrastructure and transfers the execution of the workflow on the Cloud infrastructure. To make the infrastructure as cost-effective as possible, Sunflower releases Cloud resources if the overall load returns to be normal. Sunflower has the capability of monitoring the changing system status, analyzing and controlling tradeoffs among multiple QoS features, and self-adapting its service configuration to respond to changing requirements or environment. To handle peak loads, Sunflower integrates the scheduling algorithm with a EC2-like provisioning component that can dynamically launch virtual machines in a Cloud infrastructure and deploys the required middleware components on-the-fly.

The remainder of this paper is organized as follows. Next section provides a description of the architecture for the decentralized execution of workflows in Sunflower according to choreography model. Section II describes the Cloud-enabled workflow system. Section III describes preliminary experimental results and Section IV concludes the paper.

I. SUNFLOWER FRAMEWORK

Sunflower is an adaptive P2P agent-based framework for configuring, enacting, managing and adapting autonomic workflows. Sunflower assumes that multiple copies of a Web service, with different performance profiles and distributed in different locations, co-exist. During the execution of the workflow, if a service fails or becomes overloaded, a self-reconfiguring mechanism based on a *binding adaptation* model is used to ensure that the running workflow is not interrupted but its structure is adapted in response to both internal or external changes.

Workflows are described in Sunflower by the BPEL language in order to exploit existing design tools. Sunflower replaces the standard BPEL engine with a new decentralized engine able to exploit the dynamic information available in the Grid and respond to the dynamic nature of the Grid. Figure 1 shows the architecture of the framework Sunflower. The workflow process is enacted by a set of cooperating Sunflower BPEL engines (SBE), instantiated at all participating nodes, who are responsible for interpreting and activating part of the process definition and interacting with the external resources -*invoked web services*- necessary to process the various activities.

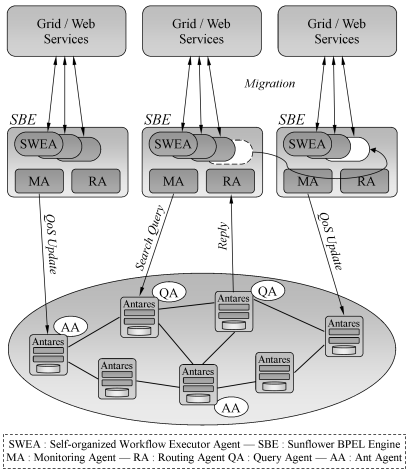

Fig. 1. The Sunflower architecture.

A dynamic group of bio-inspired mobile agents *SWEA* [5] representing the workflow executors generated from the BPEL workflow specification are initially deployed on the basis of the workflow configuration. In Sunflower the coordination model, that describes how the generated agents cooperate with each other to reach a choreography execution, is obtained by the Petri Net (PN) associated with the BPEL program [4]. The PN representation of the program is then structurally decomposed into a set of distributed sub-flow schemas. On the basis of these schemas, Sunflower enacts the federation of *SWEA* agents that will be executed on the SBE nodes. The decentralized execution of the workflow is coordinate by tokens exchanged among the SBE platforms. Tokens contain the whole execution state, including all data gathered during execution. Each *SWEA* agent performs the portion of workflow assigned and determine which agent should be activated next.

SWEA agents adapt their structure moving over the grid to position themselves in the nodes with low workload and where are available the Web services with the best performance. The framework provides support for the migration-transparent of the agents and instructs the agents, by a migration policy, to migrate in order to achieve goals like load balancing, performance optimization or guaranteeing QoS.

Sunflower monitors by Antares [6] the QoS for Web services and effectively self-adapts the workflow engines in response to changes in load patterns and server failures. Antares is able to disseminate and reorganize service descriptors by an ant clustering algorithm and, as a consequence of this, it facilitates and speeds up discovery operations. Based on dynamic service performance evaluation, the services with similar or same metric are gather by Antares into clusters. All member services in a cluster provide similar or same QoS after service clustering. Scheduling managers make scheduling decision based on user QoS requirements and information in Antares. Consequently, the task scheduling involves two steps: initial cluster selection from service clusters and further service selection from the selected cluster.

To support workflow adaptation the *SWEA* agents are assisted by routing/scheduling *RA* agents and monitoring/analizing *MA* agents that interact with the Antares information system. The MA agent collects details about the performance metrics and workload of a Grid service and when detects a change, due to external events, it inserts a new Web service descriptor with the new information in the Antares virtual space and notify the change to the RA agent. When the RA agent receives a notification about a modification of the class of QoS, it sends a query to Antares to discovery and select a descriptor of an equivalent optimal service. Then, Antares returns to RA agent a reference to an end point handler for the selected service. Before to execute the sub-workflow, the *SWEA* agent contacts the *MA* agent to verify if the class of QoS of the service to invoke is respected. In the affirmative case, the *SWEA* agent invokes the service and performs the workflow task, otherwise it uses its migration-policy to decide its destination consulting the RA agent. The activities of the *MA* and RA agents are performed continuously.

II. THE CLOUD-ENABLED WORKFLOW SYSTEM

In this section, we present the scheduling mechanism that extends Sunflower to handle peak load situations generated by excessive demands due to the simultaneously access of many researchers or customers that share web services in a Grid's virtual organization or use computational-intensive web services. To resolve workload variations generated by services with a QoS degraded or peak load situations Sunflower before attempts to use alternative grid services with the requested QoS and in case all grid services have a degraded QoS it uses additional computational resources provided on-demand by a cloud computing infrastructure. A schema of the Cloud-enabled Sunflower architecture is shown in figure 2. Sunflower

seamlessly integrates dedicated grid resources and on-demand resources provided by the Eucalyptus's Cloud infrastructure and allows to invoke Web-Grid services implemented according to the WSRF standard by GT4.

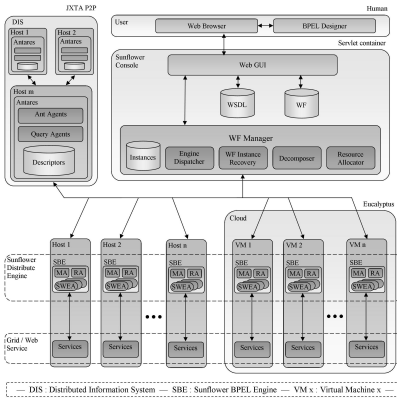

Fig. 2. The Cloud-enabled Sunflower architecture.

The Sunflower Console provides the user's interface to model a workflow using the Eclipse BPEL Designer and generate the XML-BPEL code. Through the *decomposer* module the XML-BPEL code is partitioned into sub-workflow schema that are associated to the SWEA agents. The Sunflower Console also enables the deployment of the workflow according the initial configuration on the nodes of the Grid. During this phase the SWEA agents are queued on SBE nodes that contain the Web services to invoke. In order to mask the complexity of the underlying infrastructure two different environments must be installed. A standard Sunflower environment is installed on the Grid nodes. On the Cloud, Sunflower uses a Software as a Service (*SaaS*) delivery model which provides different customers the functionality of an application that is completely hosted in the cloud. According the SaaS model, Sunflower environment is installed on a running virtual machine (VM) and then the VM is saved as an image that could be used to deploy future multiple copies of the original VM. During the runtime, if the overall load is normal, Sunflower dynamically discovers by Antares the available Web services and performs a dynamic mapping of the SWEA agents to nodes of the Grid where are available the requested services with the suitable QoS. The SWEA agents migrate from one SBE queue to another SBE run queue if the current host provides a Web service with a degraded QoS. When a peak load situation occurs the scheduling decision might involve starting new virtual machines by calling the provisioner that encapsulates interfaces to manage virtual machines in on-demand infrastructure like Amazon EC2. The provisioner provide general abstractions to upload virtual machine images, download, modify, delete, or save copies of preexisting images and deploy images as virtual machine. Sunflower uses the scheduling algorithm outlined in figure 3, executed by RA agents, to make these choices, using information provided by Antares. For each Web service, the RA agent schedules the SWEA agents queued in the local SBE.

```
1. foreach SWEA agents in queue on SBE
2. {
3.     if (SWEA.ReqService.QoS > SBE.CurrentQoS)
4.     {
5.         AlterSBE = Antares.SearchQuery(SWEA.ReqService.ServiceType,
                   SWEA.RequiredService.QoS);
6.         if(AlterSBE != null)
7.             {
8.                 SBE.Routing(SWEA, AlterSBE);
9.             }
10.            else
11.            {
12.                AlterSBE = ec2-describe-instances(SWEA.ReqService.VMDiskImage,
                           SWEA.ReqService.QoS)
13.                if(AlterSBE == null)
14.                {
15.                AlterSBE = ec2-run-instances(SWEA.ReqService.VMDiskImage,
                           SWEA.RequiredService.QoS)
16.                }
17.                SBE.Routing(SWEA, AlterSBE);
18.            }
19.        }
20. }
```

Fig. 3. The RA scheduling algorithm.

The scheduled SWEA agent checks if the QoS of the service relied on the node of the grid is less than that required. In case affirmative, a request is sent to the Antares registry service to search for an equivalent service to replace. If the service exists and is available, the reference to the service is returned to the RA agent that uses this information to migrate the SWEA agent on the node where the service is localized. Otherwise, if there are not services available an activation request of a new virtual machine is sent to the provisioner. Before that, a new VM is activated the provisioner checks if one VM with the QOS requested already exists else a new VM is started. The VM continues the execution of the workflow and before to invoke next Web service on the Cloud the RA agent checks, by querying Antares, if an equivalent service is available on the Grid. In case affirmative, the activation token with the status information is transferred to the SBE node on the grid that has the next service to invoke.

III. Experimental results

We conducted preliminary experiments considering a scenario where a local grid is at times overloaded preventing the execution of additional workflows because the web services cannot maintain the QoS required. In this case, we temporarily extend the grid thorough the use of IaaS Cloud resources to specifically allow a statically sized system to cope with an increased load. To demonstrate the operation of the framework and to evaluate its ability, we built a hybrid infrastructure with our IcarGrid and 5 Eucalyptus instances types that provide an interface similar to Amazon EC2. Each instance type is considered as a resource class. The table I describes in detail the EC2 resource classes. The performance of our scheduling algorithm has been evaluated using a geo-workflow which is apt to simulate many complex real world geo-hazard cellular automata (CA) models as landslide evolution, lava flows, floods, etc. Our sample application is a geo-workflow that

performs real-life landslide simulation related to the zone of Sarno (South Italy). More information can be found in [7]. In our experiment, multiple geo-workflows, up to seven, can be performed concurrently.

VM name	CPUs	Memory (MB)	Disk space(GB)
m1.small	1	128	2
c1.medium	1	256	5
m1.large	2	512	10
m1.xlarge	2	1024	20
c1.xlarge	4	2048	20

TABLE I
THE TYPE OF VM USED IN THE STUDY.

For our measurement, we used the *c1.medium* host type available on the Cloud infrastructure as it has shown to have the best application Time-To-Completion (TTC). Figure 4 shows the execution time in seconds of a web service for the visualization when it is invoked from 1 to 7 workflows simultaneously. Note that the scheduling algorithm allocates to the workflow the web service on the machine with the higher performance. When more of three workflows are instantiated the IcarGrid is enable to run additional workflow and up to 4 failures are generated. To continue the execution of the workflows maintaining reasonable times new virtual machines in the Cloud environment are booted.

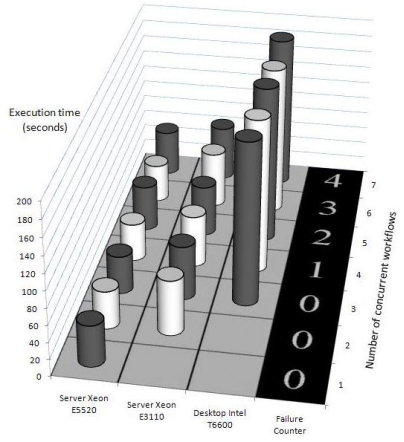

Fig. 4. Execution times of multiple workflow instances in IcarGrid .

Figure 5 shows the case when from 1 to 4 VM are booted to handle the peak load situations. Multiple workflows has been run more than 30 times. The average runtime measurements including the startup of new virtual machines are shown in table II.

IV. CONCLUSIONS

In this paper, we have presented a decentralized and cooperative strategy for the execution of autonomic workflows on heterogeneous distributed platform, such as Grids and Clouds. A bio-inspired cross-layer approach is used to support the self-adaptation of the workflow enactment engines to changes in

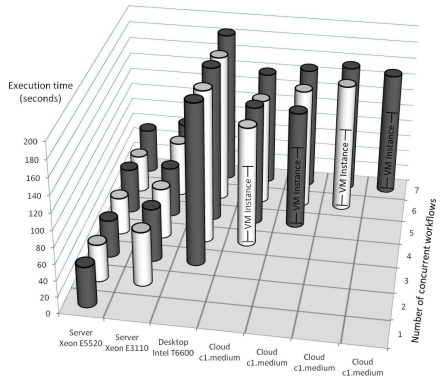

Fig. 5. Execution times for the execution of multiple workflow instances.

peer	xeon E5520	xeon E3110	intel t6600	c1.medium	c1.medium	c1.medium	c1.medium
1	49.703	-	-	-	-	-	-
2	45.828	66.578	-	-	-	-	-
3	46.047	65.39	195,953	-	-	-	-
4	45.781	62.375	186.313	146.828	-	-	-
5	54.984	61.438	189.859	146.344	143.578	-	-
6	46.109	67.531	180.25	130.546	146.578	156.578	-
7	54.954	66.484	185.781	140.546	149.39	156.438	150.578

TABLE II
EXECUTION TIMES OF MULTIPLE WORKFLOWS INCLUDING THE STARTUP OF NEW VIRTUAL MACHINES.

the Grid and fulfill QoS requirements for Web/Grid services. To handle peak loads, Sunflower integrates the scheduling algorithm with a provisioning component that can dynamically launch virtual machines in Eucalyptus's Cloud infrastructure and deploys the required middleware components on-the-fly. The overhead introduced from the Cloud infrastructure have been evaluated for the execution of a geo-workflow.

REFERENCES

[1] W. van der Aalst and K.M. van Hee: Workflow Management: Models, Methods, and Systems, *MIT Press*, Cambridge, MA, 2002.

[2] R. Buyya, Chee Shin Yeo, and S. Venugopal: Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008)*, IEEE CS Press, Los Alamitos, CA, USA, Sept. 25-27, 2008, Dalian, China. - Keynote Paper, 2008.

[3] M. Rahman and R. Buyya : An Autonomic Workflow Management System for Global Grids, *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008)*, IEEE CS Press,Los Alamitos, CA, USA, pp.578-583, 2008.

[4] G. Papuzzo, G. Spezzano: Processing Applications Composed of Web/Grid Services by Distributed Autonomic and Self-organizing Workflow Engines" in *Parallel Computing: From Multicores and GPU's to Petascale*, B. Chapman, F. Desprez, G.R. Joubert, A. Lichnewsky, F. Peters and T. Priol (Eds.), Advances in Parallel Computing, IOS Press, vol.19, pp. 195-204,2010.

[5] Brazier, F.M.T., Kephart, J.O., Van Dyke Parunak, H. Huhns M.N. : Agents and Service-Oriented Computing for Autonomic Computing: A Research Agenda, *IEEE Internet Computing*, vol. 13 n. 3, pp. 82-87 2009.

[6] A. Forestiero, C. Mastroianni, G. Spezzano: Antares: An Ant-Inspired P2P Information System for a Self-Structured Grid, *BIONETICS 2007 - 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, Budapest, Hungary, 2007.

[7] G. Folino, A. Forestiero, G. Papuzzo, G. Spezzano,: A Grid Portal for Solving Geoscience Problems using Distributed Knowledge Discovery Services", *FGCS - The International Journal of Grid Computing: Theory, Methods and Applications*, vol. 26, n. 1, pp.87-96, 2010.