Inter-Clouds: Grid Prosthesis for Workflow Systems

Reginald Cushing, Marc X. Makkes, Rudolf Strijkers, Adam Belloum
University of Amsterdam

Abstract

Nowadays cloud infrastructures have garnered popularity due to facility of outsourcing resources. Infrastructure as a Service clouds provide a flexible way to allocate remote distributed resources, but lack mechanisms to configure software dependencies and to execute and manage applications. To make most use of cloud infrastructures, scientific applications should be able to harness cloud resources from different providers to maximize its performance. Here we show how a Scientific Workflow Management Systems (SWMS) is coupled with an Inter-Cloud (IC) infrastructure and scales out grid resources onto cloud so as to scale up workflows. We present a workflow executing on top of grid and IC. We show that scaling happens at two levels: the workflow layer, where the workflow tasks are scaled independently and the resource layer where the resources are scaled on-demand to match the demand from the application.

1 Introduction

Infrastructure as a Service cloud computing enables users to dynamically create application-specific computing environments by giving users the ability to control Virtual Machines (VM) [12, 11, 9, 6]. Although still a manual process, cloud technology also enables users to federate clouds for specific applications [7]. We use the term Inter-Cloud to refer to a federation of clouds, similar to the federation of networks into the Internet and here we propose an architecture to support application execution on the Inter-Cloud. Specifically, we target Scientific Workflow Management Systems (SWMS) as applications running on the Inter-Cloud. Scientific workflows are often farmed as would be the case in parametric studies. Farming entails the acquisition of more resources to accommodate the new workflow instances. For this the IC is ideal as it allows on-demand scaling of resources. Early adopters of cloud technology show typical use cases for the work presented, such as an early warning system for the failure of dikes and other embankments and e-Science experiments in particle physics and astronomy [15, 10, 13, 2].

In Section 2 we describe the architecture of coupling the SWMS to IC. This is followed by an evaluation of the system (Section 3) using a scientific application which is scaled up by the SWMS and thus triggering a scale out in the IC. We conclude and address future work in Section 4.

2 System Overview

The System is composed of 3 main components: Datafluo, the SWMS, which acts as the workflow coordinator by coordinating task execution and communication, the Transient grid Manager (TGM) to ensure that enough memory and processing capacity is available by allocating and managing resources in the Inter-Cloud, and Elastic Site Manager (ESM) to monitor and control system software, applications, and configurations for a single or for a cluster of VMs. The latter 2 form the IC manager.
2.1 Datafluo Workflow Manager

The workflow manager, Datafluo, is a client/server run-time system developed to support dynamic use of computing resources. In the client, a user composes the workflow of a series of applications with a data flow Model of Computation. The enactment engine (Figure 1) determines the execution order of tasks with regard to data dependencies. The task auto-scaling [3, 4] handles farming (run the same computation on partitioned data) and parameter sweeps (run computations with different parameters on the same data), which require separate data dependency processing.

The resource submission scheduler manages a number of resource submitters (handlers) and maps tasks to resources. Tasks which are flagged as runnable by the enactment engine are picked up the submission scheduler and submitter to a resource through one of the submitters. The submitters allows Datafluo to be coupled with any kind or resources by plugging in the correct submitter.

Currently two resource submitters are implemented: a grid handler and TGM (Section 2.2) handler. The TGM handler monitors the status of cloud resources and when resources become available it registers the resource as a virtualized resource in the resource manager.

2.2 Transient grid Manager

The Transient grid Manager (TGM) controls allocation, creation, and removal of Transient grids (Tgrids), a temporary compute cluster with grid middleware installed, over multiple clouds. Cloud Adapters (CAadapter) encapsulate the differences between the various cloud APIs (CAPI) and regularly post status events on a message bus by polling the local CAPI. The TGM supports dynamic addition and removal of clouds, but a cloud that becomes available to the TGM has to register with the message bus through its CAadapter.

In the current architecture, there is no solution to dynamically associate a CAadapter to a TGM, because no protocols nor standards exist to discover and maintain a global information base of clouds in the Internet. However, we expect that in the future a global information base that summarizes and describes which clouds can be accessed from where will be available. Such an information base will then be used to associate clouds to the TGM.

To start a specific application the TGM selects and starts a VM from a directory of VM templates. Then, the TGM selects clouds using a cost factor. The cost factor is currently a fictional price per hour, but other cost representations are possible, e.g. latency/bandwidth to a storage provider or sensor network. After the VM is started, a control program on the VM (the Elastic Site Manager) pulls additional configuration information, such as available budget and amount of nodes, from the TGM.

2.3 Elastic Site Manager

The execution environment of an application is composed of one or more nodes. The Elastic Site Manager (ESM) runs inside the first created VM (head node) and creates, configures,
and controls local VMs. Currently the ESM implements a Transient grid (Tgrid), i.e. an on-demand compute cluster running the Globus Toolkit 4. Once the Tgrid comes up, the head node registers its initially empty job queue to the Datafluo TGM handler. Datafluo uses the empty queue to schedule (urgent) jobs and to transfer data to and from the nodes. In addition, the Tgrids run Ganglia [14] for resource monitoring and register to a central Ganglia server. We refer to previous work for further details on the implementation [1].

The ESM coupled with TGM form the core parts needed for the Inter-Cloud manager. Since cloud resources are budget driven the application making use of such resources needs to be aware of these budgets. Datafluo implements budgeting within its resource submission. The consumed budget can be retrieved from the resource handlers and the scheduler maintains budget reserves to allow graceful termination of tasks i.e. tasks terminate before the VM is shut-down because of no budget.

3 Evaluation

As a driving application we created an image processing workflow (Figure 2) using Matlab functions. The workflow is meant to exploit the scaling techniques employed by the workflow engine for dealing with farming and parameter sweeps (Figure 3).

The workflow execution uses a combination of grid and cloud resources. The latter being used to accommodate the farming capabilities of the workflow engine. The initiated on-demand cloud clusters can be seen in Figure 4.

As the Datafluo engine starts scaling the workflow modules, more resources are needed and thus TGM starts virtual clusters on private clouds.

The test bed included three clouds: a 128 core OpenNebula cloud at SARA high performance computing center, Amsterdam, The Netherlands, a 64 core intel XenServer cloud at TNO, Groningen, The Netherlands, and a 24 core XenServer cloud in New Orleans, US. The
OpenNebula cloud used KVM virtualization technology, while the XenServer clouds used Xen virtualization technology). We also used the 41 node (164 core total) Amsterdam site of the DAS-3 distributed compute cluster as a grid resource. The system software ran on two machines: a Datafluo server and a server running the TGM and Message Bus, all implemented in Java.

4 Conclusion and Future Work

In the end, applications will need control over distributed resources to deal with peak demands, failures or to optimize execution. We presented ICOS, an Inter-Cloud Operating System that controls allocation, distribution and execution of a series of applications on the Inter-Cloud, a federation of clouds. We showed that grid technology provides mechanisms to support application execution in the Inter-Clouds.

Single cloud infrastructure is currently only supporting small and medium scientific experiments that are too small for large super computers but too big for desktop systems. The Inter-Cloud and ICOS breaks this paradigm such that scientist can execute large scientific jobs and might be a good alternative to super.

Better interfaces into the ICOS would enable the application layer to have better control over the underlying infrastructure and thus paves the way for smarter scheduling such as data locality optimizations and energy aware scheduling.

We are currently working on new scientific workflows WAVE [5] (blood flow simulations) and application from the 1000 Genomes Project\(^1\). These applications are characterized by the high demand on resources this will allow use the further proof the system by scaling on both dimensions one being auto-scaling the application by monitoring data loads and second scaling the resources through the IC to match the application.

To fully realize Inter-Cloud, however, more standards and protocols are needed. For each cloud implementation different images where needed. VM standards and protocols to exchange VM between cloud providers would enable application schedulers such as Datafluo to better manage runtime tasks through migration. We consider Inter-Cloud as an Internet technology. Therefore, we find that the IETF cloud play a more prominent role in the development of the Inter-Cloud standards and protocol and policies for job management, execution and information exchange as well as an Inter-Cloud federation- and operation- framework.

\(^1\)http://www.1000genomes.org/
References


