CANFAR

Canadian Advanced Network for Astronomical Research

Cloud Scheduling System Requirements Specification

Version: 2.0
Submission Date: 2011.06.04
Prepared by: Duncan Penfold-Brown, Patrick Armstrong, Michael Paterson

University of Victoria

UBC

CANARIE

NRC·CNRC From Discovery to Innovation... De la découverte à l'innovation...

1 Revision History

Name	Date	Comments	Version
Duncan Penfold-Brown	June 2009	Initial document draft.	0.0
Duncan Penfold-Brown	26 August 2009	Revisions and formalization.	1.0
Duncan Penfold-Brown	1 September 2009	Further revisions and requirements additions.	1.0
Michael Paterson	4 July 2011	Update	2.0

2 Introduction

2.1 Purpose

The purpose of this document is to describe and formalize the requirements of the cloud scheduling system that is being developed as a part of the CANFAR processing prototype. The document will describe the cloud scheduling system and its environment, and will present a number of use cases from which the requirements of the system are derived. Constraints on system design resulting from environmental restrictions will also be described.

This document is intended for CANFAR research and project management personnel, with the goal of explaining the cloud scheduling system and providing a reference for cloud scheduler functionality. This document also serves as guide for continuing development on the cloud scheduling system.

2.2 Scope

The scope of the cloud scheduling system encompasses a large number of resources within both the UVIC and CANFAR systems. The cloud scheduler itself relies on a preexisting job submission and scheduling resource (currently, a Condor job scheduler running on the cloud at UVIC), and a number of computational clusters running cloud management software. Details of these reliances are described in following requirements.

The cloud scheduling system will interact with many job-related, cluster-related, and cloud-related entities. These are, in brief: a job scheduling system such as a Condor job scheduler; a job submission system such as the Condor client system; a monitoring and discovery service (MDS) responsible for providing up-to-date information on cloud resources; a repository providing storage for virtual machine images required for cloud scheduling; and cloud interfaces providing cloud and virtual resource management. The cloud scheduling system will use these entities to gather job requirements from submitted jobs, maintain up-to-date information on clouds visible to the system, and conduct cloud operations to provide environments for submitted jobs to run as specified.

The aim of the cloud scheduling system is to provide a scheduling system that automates the process of manually customizing an environment to suit scientific computational tasks. The cloud scheduling system provides an environment in which job environment requirements can be specified simply, and can then be submitted in the same manner by which scientists would submit a computational job to a cluster. The cloud scheduling system is then responsible for creating an environment in the cloud that will execute a large number of jobs efficiently and correctly.

2.3 Terms and Definitions

Herein:

- VM stands for virtual machine. Virtual machine is synonymous to virtual resource.
- A “cloud” refers to a collection of clusters, all of which are running grid middle-ware (such as the Globus Toolkit) as well as cloud interface software (such as the Nimbus service). These clusters are connected to a job scheduler that is responsible for distributing jobs to the cloud's resources.
- “Cloud resources” is a general term that refers to the clusters and individual computers that make up the cloud.
- The “cloud environment” refers to configuration and status of Vms in the cloud
- A job is a computational task specified in a job submission file.

2.4 Document Overview

The remainder of this document is structured as follows: The next section (section 3) contains an overview of the design and function of the cloud scheduling system. The salient features of the cloud scheduling system will be described, along with its major components and responsibilities. The characteristics of the system visible and important to the user will then be briefly described. The fourth section contains the specific requirements for the cloud scheduling system, in part derived from use cases found at the beginning of the section (section 4.1). Further requirements will be listed by category.

3 The Cloud Scheduling System

3.1 Cloud Scheduler Function and Design

The cloud scheduler is software that resides within a cloud and is responsible for creating and manipulating cloud resources in order to create an effective (fast, efficient, and correct) environment for job execution. The scheduler does this based on information on the current cloud environment (for example, the number and type of VMs already running in the cloud, and the remaining resources available for creating new VMs) and on information indicating job requirements obtained from a traditional job scheduler.

Figure 1, below, depicts the location, function, and interactions of the cloud scheduler in a cloud environment. The image is described in the following section.

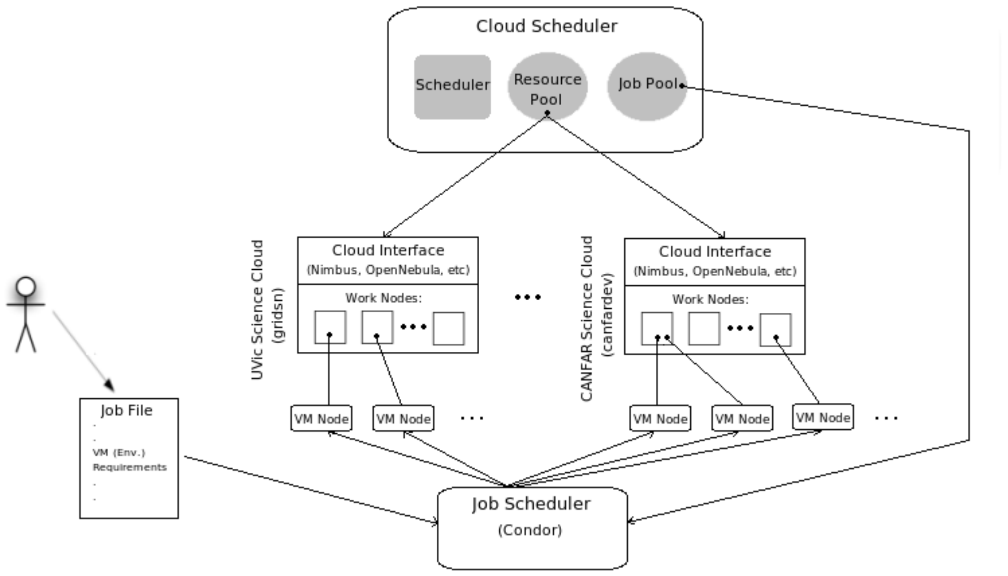

Figure 1: The Cloud Scheduler Environment

3.1.1 Function

The cloud scheduling system functions as follows: A user submits jobs to a job scheduler, which queues user jobs and is responsible for distributing them to available resources on the cloud (where available resources are VMs running in the cloud). The cloud scheduler accesses this job scheduler to obtain the job scheduler's current information, such as how many jobs it has queued, and what type and quantity of resources the queued jobs require.

The cloud scheduler also has access to information on the current status of the cloud, including what VMs are running on what physical machines in the cloud. The cloud information will be gathered by interrogating an MDS (monitoring and discovery service) that is responsible for maintaining up-to-date and accurate information on the cloud, or by reading a static configuration file containing details on available cloud systems (Note that the MDS is not pictured in Figure 1).

Based on information obtained from the job scheduler regarding current queued jobs and information on the status of cloud resources, the cloud scheduler makes decisions regarding the creation or deletion of specific cloud nodes (VMs) in order to best suit the set of jobs waiting to be run. That is, the cloud scheduler will create or destroy VMs in the cloud in order to create an efficient job execution environment, based on queued jobs and current cloud configuration.

3.1.2 Design

The cloud scheduler is made up of a number of components. These components are: the main scheduler component, responsible for coordinating all other components and for sending cloud manipulation instructions to various clusters across the cloud; the job information component, which is responsible for maintaining up-to-date information regarding jobs in the job scheduler's job queue; the cloud information component, which is responsible for maintaining up-to-date information on the current cloud environment; the resource selection component, which is responsible for matching job requirements to resource space available on the cloud for creating VMs; and the cloud management component, which is responsible for providing the main scheduler with an interface to cloud functionality on various clusters in the cloud.

An important feature of the design of the cloud scheduler is that the system is modular, and is thus structured to support multiple technologies both in the job information component and in the cloud management component. That is, the cloud scheduler is designed to be able to function with multiple cloud interface technologies, such as Nimbus, OpenNebula, Eucalyptus, and potentially other cloud interfaces. Similarly, the cloud scheduler is has the potential to function with different job schedulers, such as Sun's Grid Engine, PBS, Torque, and Condor. Currently, development of the cloud scheduler is focused on Nimbus as cloud management software and a Condor job submission and scheduling system. These technologies were chosen specifically because they are well known and because prototyping has proven them both effective and practical for initial implementation efforts.

3.2 Features

The features of the cloud scheduling system are as follows:

- The cloud scheduler will create virtual resources on the cloud that satisfy job requirements, such that jobs can be automatically executed in custom environments.
- The cloud scheduler functions entirely automatically, and requires no manual user input.
- The cloud scheduler is responsible for managing cloud resources, including responding appropriately to cloud system failure.

3.3 User Characteristics

The user of the cloud scheduling system plays a very simple role. As described, the user creates a job submission file containing requirements both for his or her job and the environment that the job must run in. The user then submits that job to the cloud's job scheduler. This is the entirety of the user's role in the cloud scheduling system – to the user, the cloud scheduler functions invisibly.

In specifying a job's environment requirements, the user much specify a pre-configured and accessible VM on which the job will be run. This allows the user to customize a VM (consisting of operating system and software) to any desired extent. This VM must be stored by the user in a cloud-accessible location (locally on the cloud, or in an image repository, for example).

4 Cloud Scheduler Requirements

4.1 Use Cases

The following use cases represent the basic functionality of the cloud scheduler - that is, the following tasks are those that will be regularly performed in the cloud scheduling system. These use cases have been created to lay out guidelines for cloud scheduler development to follow. The requirements derived from these use cases are described in the following sections.

4.1.1 Job Arrival and VM Scheduling Case

The basic sequence that the cloud scheduler will perform on the arrival and detection of a job (as submitted to a job scheduler) is as follows:

1. The job scheduler receives a job, job_X, that requires an X_VM type of virtual machine.
2. The cloud scheduler then:
    1. finds a copy of X_VM (in an image repository, or as specified by location in the job_X's job description file).
    2. asks a registry service (the cloud MDS) for the current status of cloud resources, building an internal representation of cloud resources.
    3. searches these resources for a cluster that supports job_X's VM requirements (also specified in the job description file).
    4. sends provisioning instructions (create commands via Nimbus and the workspace-control program, currently) to a selected resource.
        - this instruction creates a VM of type X_VM on the selected resource.
        - if this create call fails, the cloud scheduler will detect the failure and re-submit requests or choose new resources to submit requests to.

4.1.2 Multiple Job Arrival Case

If a job set is submitted to the job scheduler, multiple VMs of the required type (or types) may be created on the cloud. The process is the same as above, except for the following points:

- The job scheduler receives a job set, for example, 100 executions of job_Y. job_Y requires the Y_VM.
- ...
- The cloud scheduler sends provisioning instructions to the cloud to start some number of Y_VMs (the number will depend on scheduling optimization heuristics).
- ...

4.1.3 VM Termination Case

The cloud cluster will follow this sequence when a VM running on the cloud has finished executing its initial job (given to the VM by the job scheduler in the cloud scheduling system).

1. An X_VM finishes its initial job.
2. The cloud scheduler:
    1. detects that the X_VM is no longer executing a job.
    2. checks job queues for jobs requiring an X_VM type virtual machine.
        - if jobs requiring an X_VM are present in the job queues, the scheduler leaves the X_VM running.
        - if there are no jobs requiring an X_VM in the job queues, the X_VM is destroyed.

This termination case represents a very simple method of achieving VM life-cycle management. The use case for this scenario will evolve with further design efforts.

4.1.4 Specify Cloud for Job

A user may specify a target cloud or clouds for their job(s) to run on.

1. Follows the 4.1.1 case but Cloud-Scheduler is limited on what clouds to choose from.

4.1.5 Submit a High Priority Job

A job may be flagged as being high priority, such jobs should boot VMs first and receive more resources than regular jobs.

4.1.6 Re-balance Cloud Resources

When the system is currently at full or near full occupation and new user(s) submit jobs, the system must re-allocate resources so that all users have a share of the available processing.

1. A subsequent user submits jobs to the system after initial user(s).
2. The cloud scheduler:
    1. Determines a 'fair' resource distribution amongst the current users.
    2. Terminates or Retires running VMs to create space for users who are under-represented in the system.

4.2 Functional Requirements

- The cloud scheduler will communicate with a Condor job scheduler in order to retrieve information on queued jobs.
    - The cloud scheduler will potentially support other job schedulers in terms of the retrieval of job information. (Abstract, job-scheduler independent interface?)
        - DRMAA http://www.drmaa.org/ is an API for the submission and control of jobs to job schedulers. Works with Grid Engine, Condor, PBS/Torque.
    - The cloud scheduler will be able to detect when a job has finished running on a virtual resource.
- The cloud scheduler will communicate with a Condor job scheduler in order to retrieve information on available execution nodes(VMs registered with Condor)
    - The cloud scheduler will be able to detect when a VM has failed to register
    - The cloud scheduler will be able to retire an execution node
- The scheduler will be able to obtain information on clusters (that are part of the cloud). This information may include:
    - Currently running VMs, and on what clusters the VMs are running;
    - Details of currently running VMs (OS, memory, storage, processors);
    - Details of clusters, including how much available space is available for creating new VMs;
    - Etc.
- This cluster information may be obtained by the scheduler by any of the following methods (to be determined by further discussion and design):

    - By querying an MDS responsible for maintaining information on the cloud;
    - By internal storage of cluster information (i.e., by maintaining records of the scheduler's own instructions that have been sent to the cloud);
    - By independently querying the cloud resources and clusters for information;
    - By initially reading in a configuration file containing cloud information.
- The cloud may obtain cluster information by any combination of the methods described above.
    - For example, the cloud scheduler could both query an MDS for cloud information and read in an initial cloud configuration file.
- The cloud scheduler will use information on current queued jobs and on the current cloud environment to determine the VMs that should be running in the cloud.
    - The cloud scheduler will attempt to create a cloud environment that is able to facilitate the quick and efficient completion of queued jobs.
    - Specifically, the cloud scheduler will determine how many of what type of VMs should be running at what location in the cloud.
    - The cloud scheduler will send instructions to clusters on the cloud in order to create and destroy VMs (and possibly pause VMs or move VMs to different cluster locations) in order to change the cloud environment.
    - The cloud scheduler will monitor the success or failure of commands submitted to clouds, retrying or avoiding commands on non-responsive clouds.
- The cloud scheduler will be able to discover VM images in standard image repository locations given image name and other details.

4.3 Design Requirements

- The cloud scheduler will be built into a VM running in the cloud.
    - This VM will have necessary cloud interface client software installed.
    - This VM will have necessary job scheduler client interface software installed.
- The initial scheduler prototype will be built in Python, for its “simple-yet-powerful” philosophy, flexibility, and suitability for prototyping.
- The cloud scheduler will initially use local commands to query the job scheduler to obtain job information.
- The cloud scheduler will initially use local commands to manipulate workspaces on the cloud (that is, the scheduler must be designed to work with local cloud commands).
- The cloud scheduler will be composed of modules with clear functionality, which could be replaced by modules with identical functionality but different implementation.

4.4 Client Requirements

- The cloud scheduler should make monitoring information available, ideally as JSON formatted data.
- The client, or user, of the system will not be required to manually configure or manipulate the cloud scheduler.
- The client, or user, of the system will be responsible for including environment requirement data in his or her job submission files.
    - The user of the system will be responsible for the correctness of submitted requirement data in job description files.

4.5 Performance Requirements

- The cloud scheduler should be able to run continuously for at least one whole day (24 hours) at a time.
    - This is a minimum requirement: with development, the cloud scheduler should be able to run for many days at a time.