Data-intensive Workflow Execution using Distributed Compute Resources

Ashish Pandey, Songjie Wang, Prasad Calyam
Email: apfd6@mail.missouri.edu, {wangso, calyamp}@missouri.edu; University of Missouri-Columbia, USA.

***Abstract*—Cloud computing has become a necessary utility for scientific and technical applications. Many diverse web services are published and subscribed using cloud data centers. It has become fairly easy to use services from Cloud Service Providers (CSPs) for computation and data processing. However, even with all their benefits, commercial cloud resources are not economical when large data processing is required. Hence, educators and researchers need guidance to use commercial cloud resources to run large data processing workflow applications within a budget. In this paper, we propose a framework to help users to leverage distributed compute resources to execute data-intensive application workflows, under budget constraints. We demonstrate how our framework can be used by users who may have access to small-scale compute resources in-house, to seamlessly interoperate with public cloud resources.**

***Index Terms*—Compute resource utilization, Scientific workflows, distributed computing**

I. Introduction

Scientific and technological applications are becoming increasingly data and compute-intensive. An exemplar case in scientific workflows can be seen in bioinformatics such as gene sequencing/analytics, where applications frequently require diverse multi-cloud resources to execute job flows. Researchers who create these workflow pipelines seek to use large compute and memory resources on a routine basis in an iterative and repeatable manner. Since the nature of research is often iterative and requires repeated workflow execution with varying data, they need to rely on open/commercial cloud resources for their workflows under budget limitations.

To optimally fulfill user requirements, they seek to seamlessly interoperate with any compute resources they can access. A potential scenario could involve the utilization of small-scale resources in-house in conjunction with community cloud resources such as GENI [1] and commercial CSPs such as Amazon Web Services. However, the diversity in the resources offered from different CSPs and a sparsely documented in-house compute resource configurations can overwhelm users who are not well versed in cyberinfrastructure or are not cloud-experts.

In this paper, we describe a brokering framework prototype that we developed to coordinate application workflow execution by efficiently utilizing a distributed set of commercial/community cloud as well as local computing resources. Our ultimate goal is to seamlessly add any available resources using the framework for executing unique/complex compute and data-intensive application workflows.

This work was supported by the National Science Foundation under award number OAC-1827177. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

978-1-7281-2700-2/19/$31.00 2019 © IEEE

II. Motivation

For demonstrating potential use of our framework to non-expert cloud users, we created a set of tool configurations as shown in Figure 1. To execute application workflows, we collect user specified resource requirements relevant to their application through a web portal viz., the CyNeuro portal [2]. CyNeuro is a science gateway, where multiple scientific computational workflows can be deployed and executed for neuroscience researchers/educators. We integrated a questionnaire interface on the CyNeuro portal, which presents users with a set of questions, such as required processing units, minimum memory, and preferred size of storage. Using the user specifications, a distributed set of compute resources are provisioned by our brokering framework. These resources could be from commercial CSPs, community CSPs, local resources, or a mixture of them for execution of different jobs in the workflow. Once resource allocation is done, users can then configure and run their application workflows on the allocated resources. During job execution, performance of the application and the status of the deployed resources can be monitored at any time. Figure 2 shows the steps of the collection, provisioning, consumption and monitoring that we have designed for supporting a custom scientific workflow.

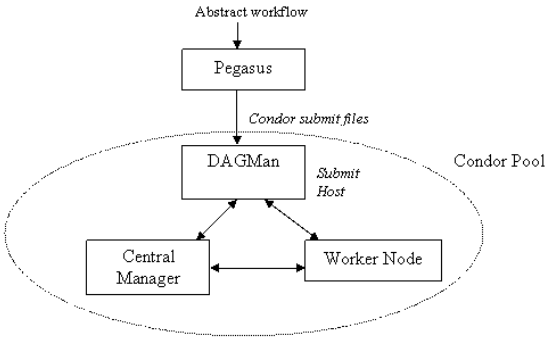

Fig. 1. Overview showing abstract workflow created by an user using Pegasus and a job processing pipeline with HTCondor.

III. Framework Components

Our framework features three main components, i.e., the resource broker, the CyNeuro user portal, and the CyVerse data portal. These components work in synergy to help users to create workflows and select the most suitable candidate cloud resources to run their workflows.

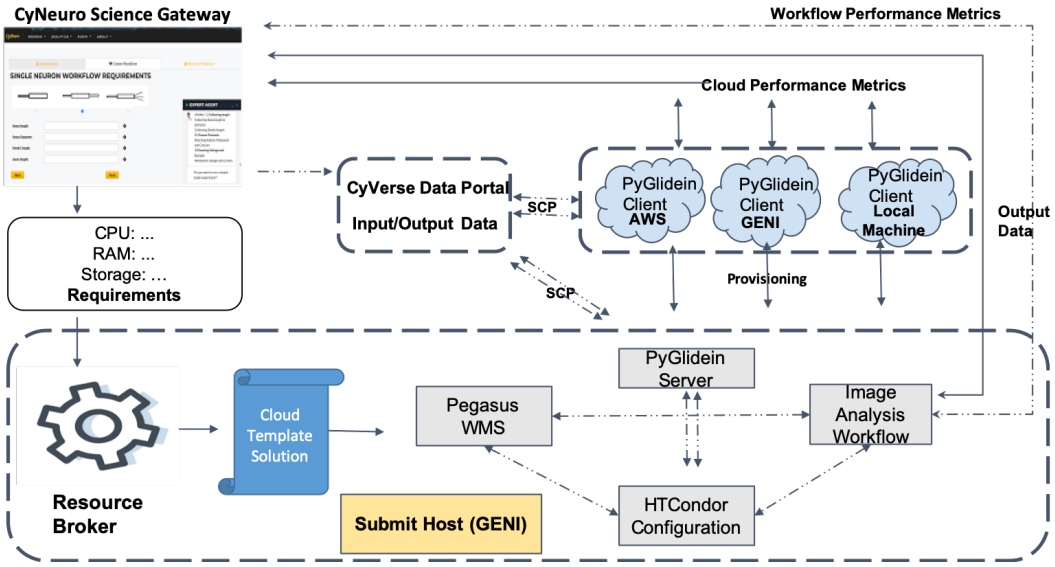

Fig. 2. Architecture design and implementation steps for representing our resource brokering framework components and their interactions.

CyNeuro User Portal: The user portal for neuroscience researchers to submit computing jobs with their anticipated required computing resources and budget availability.

Resource Broker: The computing jobs along with the computing resource requirements are collected by the resource broker component. The resource broker is based on linear and combinatorial programming focusing on minimizing cost, and uses a knowledge base of available resources from CSPs that we created. The user specified requirements are treated as constraints for the brokering. Workflows are configured in real-time to be executed on candidate CSPs suggested from our brokering using the open Pegasus/HTCondor/PyGlidein software stacks. Pegasus [3] is a Workflow Management System (WMS) that can manage large-scale scientific workflows across desktops and computing sites. It provides a means for representing the workflow of an application in an abstract form compiled into an executable form that can be deployed on a local and/or remote distributed resources by HTCondor [4]. HTCondor is a high-throughput computing software for coarse-grained distributed parallelization of computationally-intensive tasks. Pyglidein [5] is used to run *glideins* on remote sites. It consists of a server running on the central HTCondor submit machine and a number of clients on remote submit machines. The client will submit *glideins*, which invokes a central HTCondor machine that advertises slots for the jobs.

CyVerse Data Portal: CyVerse [6] provides infrastructure for storage of large data and provides facility of accessing and downloading data on-demand. For demo purposes, we host our application data on the CyVerse storage system.

IV. Workflow Execution

To demonstrate the effectiveness of our proposed brokering framework, a GENI node is instantiated as the submit host machine. Pegasus, HTCondor and CyVerse iCommand [6] are installed on the submit host. Further, an image analysis workflow is created in compliance with the Pegasus guidelines. Using Pegasus, our image analysis workflow is configured to run on specific set of machines identified by their IP addresses.

Pegasus essentially divides the workflow into subtasks while maintaining communication links between the tasks, and the user can pre-configure individual subtasks to be run on specified compute nodes. Once the workflow is initialized by the user, Pegasus broadcasts the jobs to HTCondor to schedule them on compute resources. Depending on the combined configuration of the workflow and HTCondor, the jobs are computed locally or on specified compute machines. HTCondor uses *glidein* services from PyGlidein to connect to remote machines. The *glideins* are submitted to the remote cluster scheduler, and once started up, users perceive that their HTCondor pool extends into a remote cluster. HTCondor can then schedule and monitor the jobs to the remote compute node(s) in the same way it currently schedules jobs on local compute nodes i.e., through the use of the Pegasus APIs.

As part of our demonstration, our novel resource brokering prototype configuration allows for the execution of data-intensive application workflows with complex computing demands on remote sites such as AWS, GENI, or local organizational resources. Input data is fetched from CyVerse in real-time by the workflow and output data is generated and sent back to the Cyverse storage. Additional compute nodes could be added in real-time to the compute pool to help bioinformatics users (i.e., educators and researchers) process large datasets in a cost-effective and user-friendly manner.

References

[1] Geni, Exploring Networks of The Future. Available: https://www.geni.net/ [Online][Last Accessed: 08.09.2019]

[2] CyNeuro Science Gateway. Available: http://cyneuro.org/ [Online][Last Accessed: 08.09.2019]

[3] Pegasus workflow management system Available: https://pegasus.isi.edu/ [Online][Last Accessed: 08.09.2019]

[4] HTCondor Available: https://research.cs.wisc.edu/htcondor/index.html [Online][Last Accessed: 08.09.2019]

[5] Pyglidein Glidein Service. [Online][Last Accessed: 08.09.2019] Available: https://github.com/WIPACrepo/pyglidein

[6] CyVerse. Available: https://www.cyverse.org/ [Online][Last Accessed: 08.09.2019]

[7] AWS, Availa: https://aws.amazon.com/ [Online, Accessed: 08.09.2019]