Apache Airavata Resource Allocation System

A Tool for Allocating Resources in Science Gateways

Harsha Phulwani
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
haphulwa@iu.edu

Madrina Thapa
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
mthapa@indiana.edu

Suresh Marru
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
smarru@iu.edu

Marlon Pierce
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
marpierc@iu.edu

Sudhakar Pamidighantam
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
pamidigs@iu.edu

Marcus Christie
Science Gateways Research Center
Indiana University
Bloomington, IN, USA
machrist@iu.edu

ABSTRACT

Science Gateways provide user environments and a set of supporting services that help researchers make effective and enhanced use of a diverse set of computing, storage, and related resources. In a software framework like Airavata, the distributed computing resources such as local clusters, supercomputers, computational grids, and computing clouds are shared among multiple researchers. Hence it requires an allocations process to track the demand for their resources, understand the scientific objectives of their users, and decide among competing needs when faced with user demand that is greater than the available resources can supply. We describe the design and a prototype implementation in this presentation.

PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-6446-1/18/07.
https://doi.org/10.1145/3219104.3229271

CSS CONCEPTS

Computational service allocation

•Software and its engineering→Software design engineering;

KEYWORDS

Apache Airavata, Science Gateways, SEAGrid, Computational Services, Resource Allocation, Resource Sharing

1 INTRODUCTION

Science gateways provide access to advanced resources for science and engineering researchers, educators, and students. Through science gateways, broad communities of researchers can access diverse resources which can save both time and money for themselves and their institutions[1]. One of the important services provided by the science gateways includes computational services i.e. access to supercomputers, cloud computing, and clusters. Thus it is imperative to have a resource allocation system in any science gateway. Resource allocation system software component is mainly responsible for keeping track of requests submitted by the computational service users and also track of its usage.

A key missing component in Apache Airavata today is to give science gateway administrators the ability to assign and manage allocations to their users to make sure that certain users do not use up the gateway's entire

computational services. The Resource Allocation Manager enables tenant gateways to define allocation policies and available resources. Users of gateways (typically researchers and educators) can request appropriate resources to accomplish their research objectives by justifying their need for the resource(computational service). There are two types of allocations possible - community and campus. In this paper, we are focussed on community allocation type only. Users who request for computational services can be divided into two broad categories. Users with allocations on a resource i.e resource in third-party organizations such as XSEDE Scholars (PIs) or with exclusive access to resources example personal laptop's, server etc. Users without any allocation are the usually the ones interested in community allocation type thus we will be considering only them in this paper.

2 Need for resource allocation system in a science Gateway

In the current production deployment of Apache Airavata there is no way to track the computational usage on the resources deployed. With the implementation of an allocation manager, all resource usages can be monitored and tracked. This implementation would allow users to request for resources and justify the usage. Then, the request will enter a process a pipeline where reviewers will review the request, comment about it and the gateway administrator can accept or reject the request and its reviews and finally provide the allocation. A transparent user interface for this process reduces the dependencies on notification system or emails. Also, having defined seperate user roles and different screens for each user type keeps the process simple and clutter-free. The administrator will benefit from a comprehensive view that provides allocations and users that have access to them at the for available resources in the interface itself. Once the allocation is approved, the PI can then create groups for that allocation. The same tracking of resources can then be applied at the group level. This will ensure that one particular person is not hogging all the resources. Having an allocation manager where we know how much each user used can help the administration to gather more resources or reduce which will keep them prepared for future changes. Each user can also be charged based on only how much he used instead of a fixed amount. This Resource Management System has a application in validation of tasks for allocation and charging and preparing automated workflows and critical for resource management for NSF supported open academic resources and commercial resources and clouds.

3 Proposed solution

Resource Allocation system's main usage is to make sure that every user of the gateway gets the resource rather than few users using up the entire resource. The most important step was establishing a security model. There are three roles i.e admin, reviewer, and PI(principal investigator) (Refer Table 1) defined to facilitate the allocation process.

Security Roles	Access Rights
Admin	View all the request submitted by the PIs, reviews submitted by the reviewers for a request and finally approve or reject a request.
Reviewer	View only the requests assigned to him/her for review.
Principal Investigator	View only his/her requests.

Table1. Security model used in Resource Allocation

The basic workflow of the Resource Allocation system would be that the PI would submit a resource request with all the required details. If the requested allocation units are less than a certain threshold small value defined in the gateway, then the request gets approved automatically. This provides new users quick access to the resources for testing features and the gives an idea how the gateway works. If the request is for larger allocation the administrator assigns reviewers to assess the request. Reviewers assigned provide their reviews and the administrator approves or reject the request based on the reviews and the availability of the resources. (See figure 1).

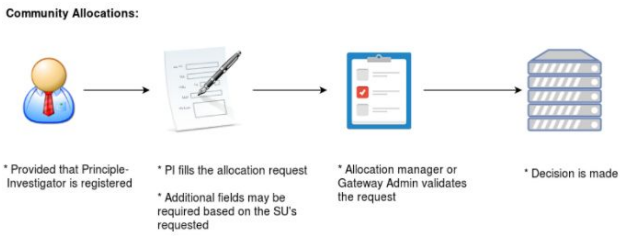

Figure 1: Basic workflow of Resource Allocation System.

Resource Allocation system was planned to be implemented as a microservice so that it could be seamlessly integrated with Apache Airavata. The Resource Allocation system leverages authentication and authorization interface developed for the Apache Airavata gateway. The User interface which communicated with the Resource Allocation microservice is developed in Django which is in coherence with Apache Airavata Django gateway. Resource Allocation in itself will act as a microservice exposing Thrift RPC endpoints for the other Airavata Django user interfaces to consume. The User Detail callback is a utility which is used to fetch the User details from Django gateway database (See figure 2).

The idea was to keep everything modular so that in future it is easy to maintain and scale. After the request for the resource has been submitted reviewed, accepted, and

an allocation is created, it is necessary to track its usage too. The Allocation Manager system which is yet another microservice will keep track of the service units used out of the total service units allocated to a PI/user.

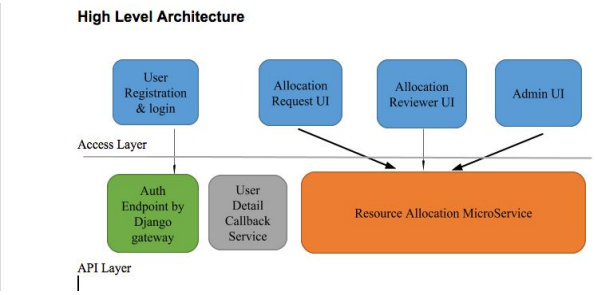

Figure 2: Example of a figure caption. (figure caption Demonstrates the use of existing Authentication and Authorization interfaces provided by Apache Airavata-Django-Portal.

4 DESIGN AND CORE SERVICES

After considering all the use cases of the Resource Allocation system in a software framework like Airavata, we implemented the architecture discussed above in the Proposed Solution section. The choice to use Apache Thrift was made to expedite the development and implementation of efficient and scalable backend services. Apache Thrift forms a Remote procedure call (RPC) and enables efficient and reliable communication across various programming languages which makes it easy to plug it in Apache Airavata Django Portal [2]. Object-Relational Mapping enables the underlying databases such as Derby and MySQL to be plugged in seamlessly to Apache Airavata. Apache OpenJPA was used as ORM solution to as it simplifies storing objects in the database [3].

Each request submitted by the user has general information regarding the request along with the specific resource request details like applications to be used, specific resources for that application and service units requested on each of the resources. We created two database tables UserAllocationDetail and UserSpecificResourceDetail to save general information regarding the request submitted by the user and specific resource details respectively. When a new request is submitted by the PI, an entry is made in the UserAllocationDetail table and a unique id (Allocation Project Id) is generated and which then is used to save details about the specific resource associated with the request. Each PI can submit only one request per year, in case there is a need for an additional resources he/she should create a new specific resource to be added to the existing request.

Once the request for resources has been submitted by the PI, admin assigns the reviewers to review the request. The reviewers can view only the requests assigned to them and can add their reviews using the forms available in the RAS. Reviewer's review is stored in the same fashion as the PI's request in two different tables - ReviewerAllocationDetails and ReviewerSpecificResourceDetails. Once the reviewers have added their reviews, admin can view them simultaneously and then decide to approve or reject the request at each of the individual resources. When an admin decides to assign allocation to a request he/she is supposed to specify the service units (amount) to be allocated and this number should be validated and never be greater than the total number of available units on that specific resource. As a request might have more than one resource associated with it unique status for and each resource i.e. pending, approved or rejected, will be available to be set and provided to the PI. The status of the request is cumulative of all the sub statuses of the specific resources that constitutes it. For example, if the request has three specific resources and all are approved then the final status of the request is also approved. If all the three specific resources are rejected then the final status of the request is also rejected. Likewise, if all any of three specific resources are approved and others could be pending or rejected then the final status of the request would be partially approved. (Refer table 2)

Specific Resource 1 substatus	*Specific Resource 2 substatus*	*Specific Resource 3 substatus*	*Final Request status*
PENDING	PENDING	PENDING	PENDING
APPROVED	APPROVED	APPROVED	APPROVED
REJECTED	REJECTED	REJECTED	REJECTED
APPROVED	REJECTED	PENDING	PARTIALLY APPROVED

Table2. Specific resource substatus and final request status

5 Implementation

We implemented the interfaces in a prototype Django version of SEAGrid science gateway. We started off by creating 3 groups in an existing django.seagrid.org organization using the group management utilities, - User group, reviewer group, admin group. We then used keycloak authentication to decide which user is logged in and which group he belongs to. After the group is identified, we loaded the screens according to each role.

User: If a user is signed in then, a user can create requests, view all requests, edit any request and view status of his request. Please refer to user screens below for more detail (Figure 3 and 4).

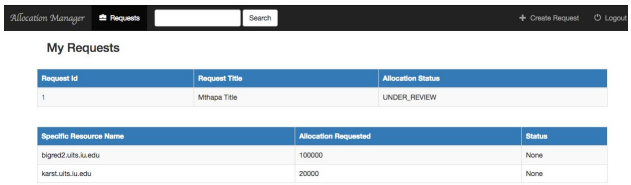

Figure 3: User(PI) Dashboard

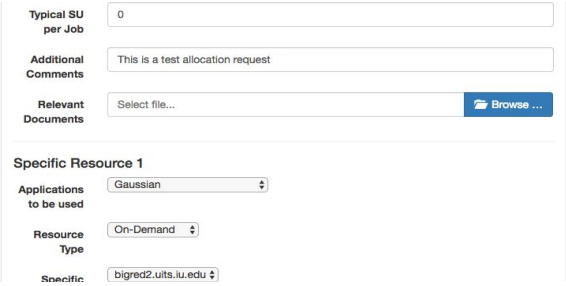

Figure 4: New request create/edit form

Admin: If an admin is signed in, an admin can view all the requests, assign reviewers to the pending request, view reviewers comments about the request and then approve/reject the request. Please refer to admin screens below for more detail (Refer figure 5 and 6)

Figure 5: Admin Dashboard

Reviewer: If a reviewer is signed then, then he can view all requests assigned to him and he will have a side-by-side view of what the user submitted, and what he has to comment about each field. She then has the option to save the review. Please refer to reviewer screens for more detail (see figure 7).

6 FUTURE WORK

In terms of future work, group integration is one of the most important things which will enable an allocation to be divided among groups. This will enable tracking the allocation not only for a single PI but for a whole group that the PI assigns. After the allocation is approved, there would be a separate option for the PI to create a new group, add or remove people from that group, and divide the allocation amongst different people. Also, this application mainly does the work of assigning allocation but enforcing allocation is not addressed here. Currently, when a user submits an experiment, there is no validation happening to check if he has resources available to complete the experiment. Our API can be called in one of the Helix workflows deployed in Apache Airavata before the the experiments are launched to check if enough allocation is available. And once the experiment finishes, the workflow should also call our API to update the allocation used for the experiment. Then, at each step of the allocation processing pipeline from user submitting the request, reviewers getting assigned, reviewers submitting feedback, admin accepting/rejecting, users should be notified through email, a notification manager has to be integrated . It would also make sense then to mine the data about each user and its usage patterns. How much each user is using, at what dates is she most active.This would make the administration be prepared for a better resource management.

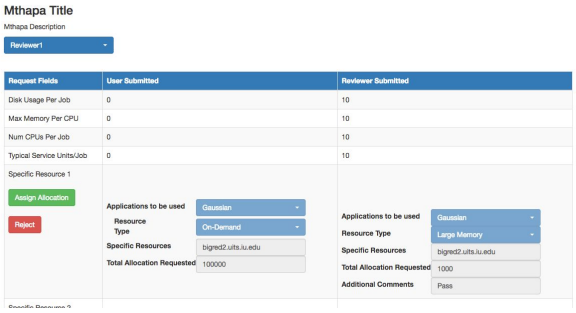

Figure 6: Admin's Request view to compare user and reviewer submitted details

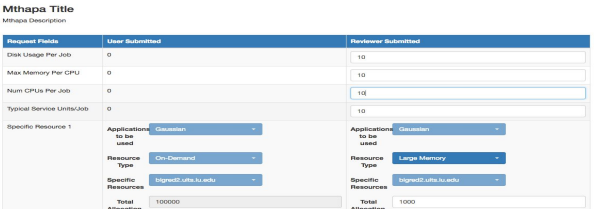

Figure 7: Reviewer's request view - form to add review for a request

7 CONCLUSIONS

In this paper we have presented a generic design for the implementation of allocation manager in Science Gateways. As the adoption of gateways accelerates, more and more users will request for resources and having a way to enforce this allocation will provide optimal use of computational resources. The goal was to create an application using such an architecture which would be easily expandable using Apache Thrift and Django framework. The paper discusses the need for such an application, its underlying architecture, data models used, API functionality and the workflow used for this application. We further described the implementation of the application in detail, keycloak authentication and group based authorization.

REFERENCES

[1] https://en.wikipedia.org/wiki/Science_gateway.
[2] https://en.wikipedia.org/wiki/Apache_Thrift.
[3] https://en.wikipedia.org/wiki/Apache_OpenJP