CSCI-B 649 Topics in Systems: Science Gateway Architectures

Class Schedule: Tuesdays and Thursdays; 4 pm to 5.15pm; I2 (Informatics East); Room 150

Office Hours: Tuesdays and Thursdays from 3 pm to 4pm; I2 (Informatics East); Room 226B

Course Website - http://courses.airavata.org/

Instructors

The course will be taught by Marlon Pierce and Suresh Marru, who lead the Science Gateways PTI Research Center and are project management committee members for the Apache Airavata open source software.

Course Overview

Science gateways are distributed computing environments that enable scientists to conduct computational experiments on computing clouds and supercomputers and have revolutionized bioinformatics, computational chemistry, nano-engineering, and other scientific fields by bringing unprecedented computing power to a broad community of scientists. Gateways are interesting topics in their own right. Modern gateway systems utilize microservice architectures and DevOps principles in their design and operations, adopting lessons learned from cloud-based Software as a Service activities.

Many gateways are also investigating how to integrate Apache's "big data" and cloud computing software projects like Apache Mesos, Apache Spark, Apache Samza, Apache JClouds and Apache Kafka. RPC versus message-oriented middleware at scale is an open question, as are NoSQL versus Relational DB approaches for gateway data management. Finally, as gateways front ends are Web-based user environments, choosing the right technologies and crafting the correct user experience are challenging problems.

In this course, students will be divided into development teams, and each team will build a science gateway software as a service system from scratch. Teams will be encouraged to explore alternative technologies and ways for building science gateways as well as learning DevOps principles such as containerization, continuous integration, and continuous deployment for deploying robust cloud services. Students will also be introduced to the Apache Software Foundation's open community governance principles for open source software and will learn how to effectively interact with Apache Software Foundation projects in order to become committers and project management committee members.

Course Objectives

- Provide a high level, broad understanding of the application of core distributed

computing systems concepts to “Software as a Service” systems that support scientific research and education.
- Study both abstract concepts and practical techniques for building science gateways.
- Provide hands-on experience in developing a science gateway while working with open source philosophies modelled after Apache Software Foundation.
- Apply the general concepts of Distributed Systems and understanding state of the art in applicable areas.

Course Outcomes

- Demonstrate an applied understanding of microservice architectures and their underlying distributed systems foundations.
- Demonstrate an applied understanding of the DevOps principles of continuous integration and delivery to the development and operations of science
- Demonstrate an understanding of open source practices, particularly those of the Apache Software Foundation.
- Demonstrate an ability to develop remote job submission interfaces to computational cyberinfrastructure like IU Big Red 2 Supercomputers.
- Demonstrate an ability to develop a simple metadata management system.
- Demonstrate an ability to develop and consume API services.

Grading Overview

Students will be divided into teams. Each team will have a project with 4 intermediate milestones. There will also be a midterm and final presentation. The maximum number of points for the semester is 100. 90-100 points is an A, 80-89 points is a B, etc.

- Course Project 80%: There will be 4 project milestones, including mid-term and final milestones. Each project milestone is worth 20 points.
    - Must use Apache compatible open source licensed software and tools
    - Projects must be checked into github, must be reproducibly executable on the deadline day by the instructors.
        - Linux/Unix compatible
        - If the instructors cannot execute your project and verify you have met the success criteria, the team receives 0 points.
        - A team may resubmit their assignment at any time before the next milestone. Each resubmission gets -1 points; i.e., 9 points if you get it right on the second try, 8 points on the third try, etc.
    - Students who show no activity (no github commits, no email discussions, etc) for the milestone will receive 0 points.
- Midterm Presentation: 5% All team members must participate. This is worth 5 points.
- Final Presentation: 5% All team members must participate. This is worth 5 points.
- Classroom Interactions and Peer Reviews: 10%. The projects and the topics will require interactive pro-active participation. Also mimicking real-world open source and software

development practices, the course requires students to be aware of other approaches to problems, borrowing ideas (with proper acknowledgements and no stealing and plagiarizing) and peer reviewing and offering constructive feedback. These demonstrated interactions (on github issues and pull requests) will be worth 10 points.

- 1 point perfect attendance
- Up to 4 points classroom interactions
- Up to 5 points for GitHub interactions with other projects (not your team's project). Examples include
  - Posting bugs that get resolved
  - Resolving bugs in other team's projects. These must be accepted to the code base.
  - Trivial issues don't get rewarded.

Project Grading

- Each project will be judged on ~4 quality attributes. The number will vary by assignment
- To get all points, the project must demonstrate all attributes to the grader. The grader must also be able to easily install and test all software by following documentation for the milestone in the team's GitHub Wiki.
- Each student on the team will submit a report describing what they did
  - Give a percentage of effort for each attribute
  - Provide auditable proof via links to issues and commits.

Project Milestones

- Project 1: Microservices with DevOps (Due September 22)
  - Microservices
    - Need a reasonable dummy use case
    - Have a local data store
    - Has a remote invocation
  - Continuous Integration and Deployment
  - APIs: REST, Thrift, Swagger, etc
  - Sample Web Interface
  - Tools & Technologies
    - At least 3 programming languages (one for the UI, at least 2 distinct ones for the server components).
    - Amazon EC2
    - Any database
    - Apache Thrift or Apache Wink for Service Interfaces.
    - Travis-CI for Continuous Integration
    - Amazon Codedeploy for Continuous Deployment
- Project 2: Security, Auditing, Distributed Coordination (Due October 20)
  - Security: OpenID Connect and OAuth
  - Inter-service communication

    - Containerization
    - Multilevel testing
    - Logging
- Project 3: Reliability & Scaling (Due November 17)
    - Fault tolerance
    - Load Balancing
    - Continuous upgrades without downtime
- Project 4: Cyberinfrastructure (Due December 8)
    - Interacting with remote computing resources and data
    - Searching, sharing metadata
    - Gateway metadata and provenance

Weekly lecture Schedule and other updates on course website - http://courses.airavata.org/