Advanced Science Gateway Architectures

Marlon Pierce, Suresh Marru

Science Gateways Research Center

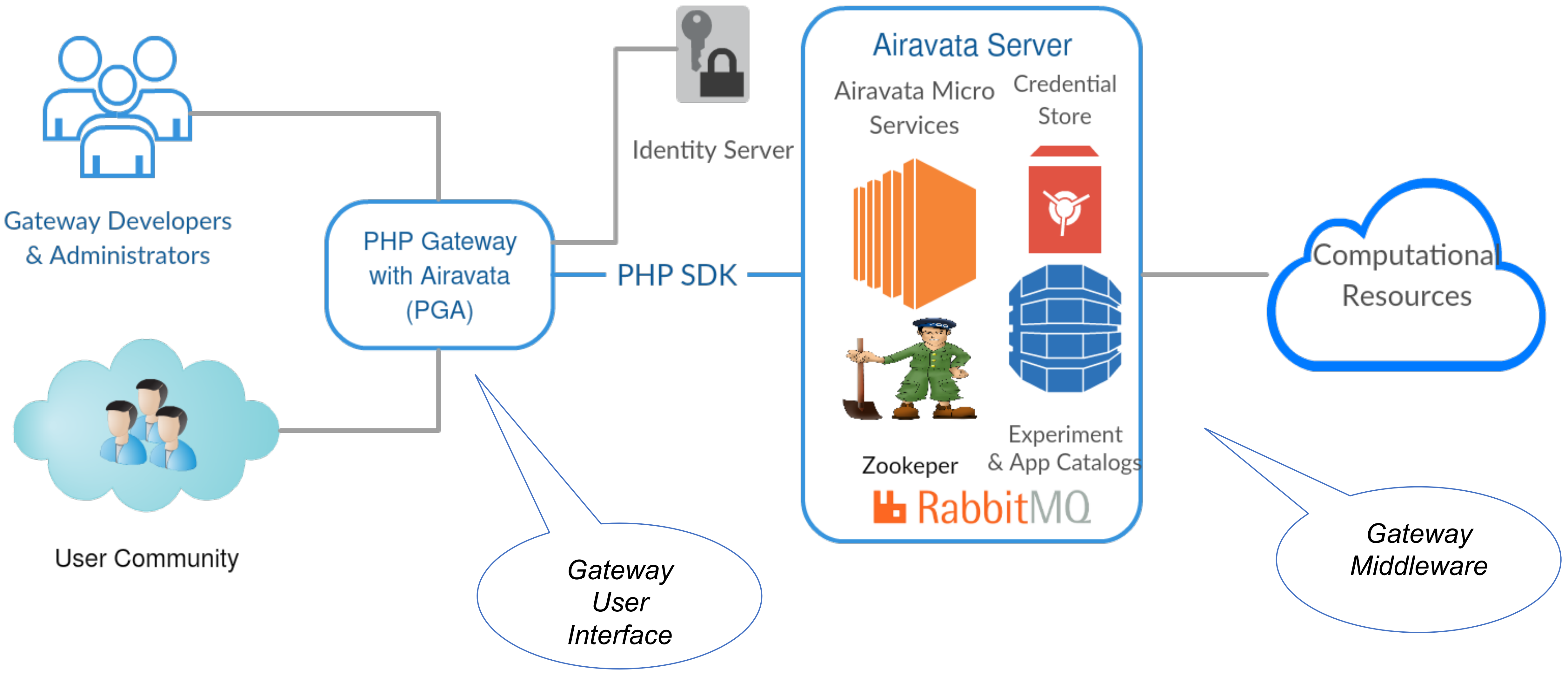
Office Hours (To Be Finalized)

- Wednesdays 10:00 am – 12:00 noon in CIB Multipurpose Room, as available.
  - This is Science Gateway Research Center’s regular team hackathon time.
- Fridays 3:30-5:00 pm
- Other times by appointment
- Course HipChat: https://www.hipchat.com/gMdyarVIz

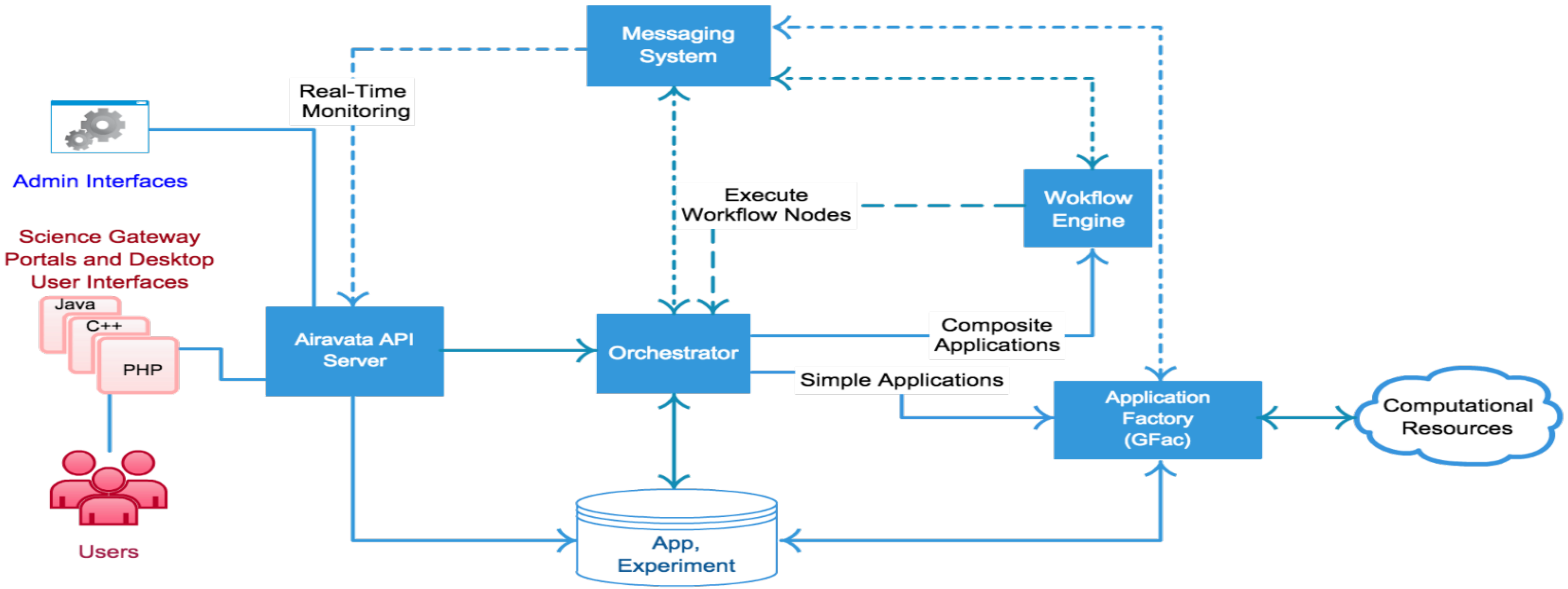
What is a Science Gateway?

Demo of PGA

Gateway Anatomy mapped to Airavata

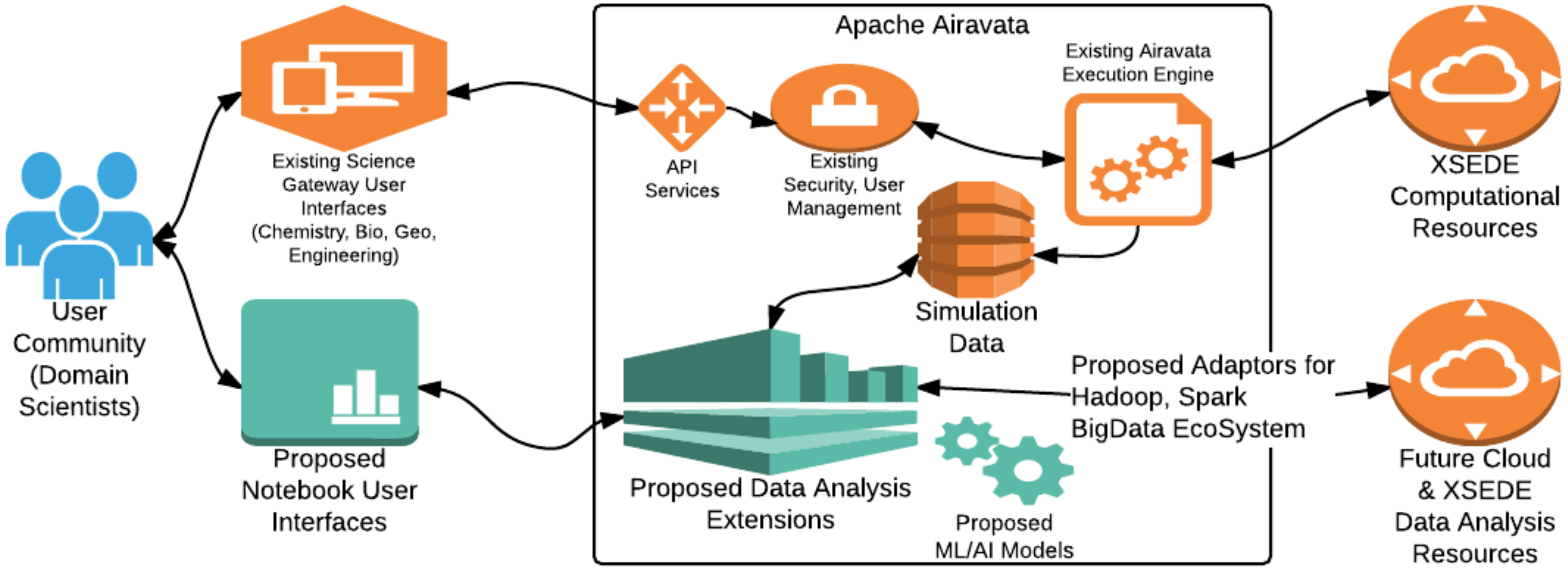


Airavata: Keep it simple, yet flexible



- External clients interact with Airavata API (based on Apache Thrift).
- Internally, components interact with each other through Component Programming Interfaces (thrift based CPIs).

Managing Computations → Data Analysis


Apache Airavata
Existing Science Gateway User Interfaces (Chemistry, Bio, Geo, Engineering)
API Services
Existing Security, User Management
Existing Airavata Execution Engine
XSEDE Computational Resources
User Community (Domain Scientists)
Simulation Data
Proposed Notebook User Interfaces
Proposed Data Analysis Extensions
Proposed ML/AI Models
Proposed Adaptors for Hadoop, Spark BigData EcoSystem
Future Cloud & XSEDE Data Analysis Resources

Fall 2016 Class Summary

- Science gateways are distributed systems on top of distributed systems.
  - Gateways talk to supercomputers, clusters, clouds
  - Gateways themselves have multiple services.
- The microservice architecture is an important (but not the only) way to build gateways.
- Microservices depend on continuous integration and delivery techniques
  - Testing, updates, etc

Fine words, but how do you really implement this?

Some Unanswered Microservice Questions

- How should microservice communications be coordinated?
- Are all microservices equal?
  - Or do we have core services?
  - How do non-core services interact with core services?
  - Are core services read-only, or can non-core services write?
- For new requirements, should I modify existing services or should I create a new set for that specific use case?
- Where is the state of an experiment?
  - Key to load balancing, fault tolerance, etc.

Class themes are based on fundamental Airavata architectural issues.

Goal of the Class

- Students should learn core distributed systems concepts by contributing to Apache Airavata.
- We are all self-interested
  - You should use this course to expand your skills, demonstrate problem defining as well as problem solving skills, learn open source governance, and make concrete contributions to a top level Apache Software Foundation project.
  - We want to make Apache Airavata better. This may mean exploring some options that don’t work out or only provide information.
- **“Contribute” means more than code.**
  - We want you to active discuss your work on the Apache Airavata developers’ and architects’ email lists.

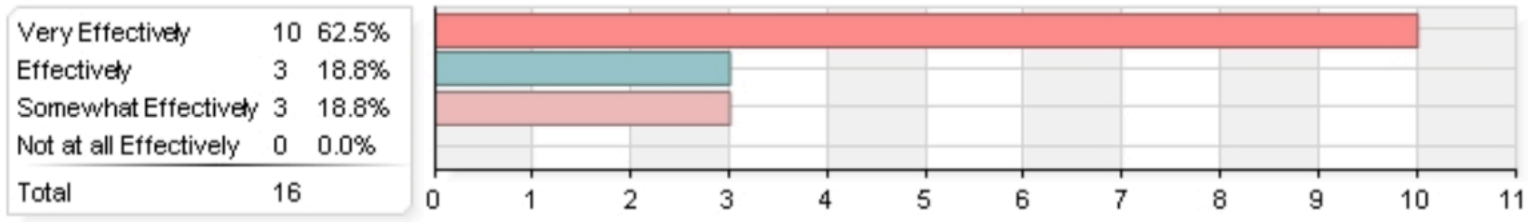
A Note on Contributing Code

- This project will not initially focus on Apache Airavata’s current code base.
  - We don’t want to bias the class with some historical implementation decisions.
- But we will provide real Airavata problems for you to investigate.
- You will work in GitHub
- You can make your code contributions to Apache Airavata.

Expected Effort Level

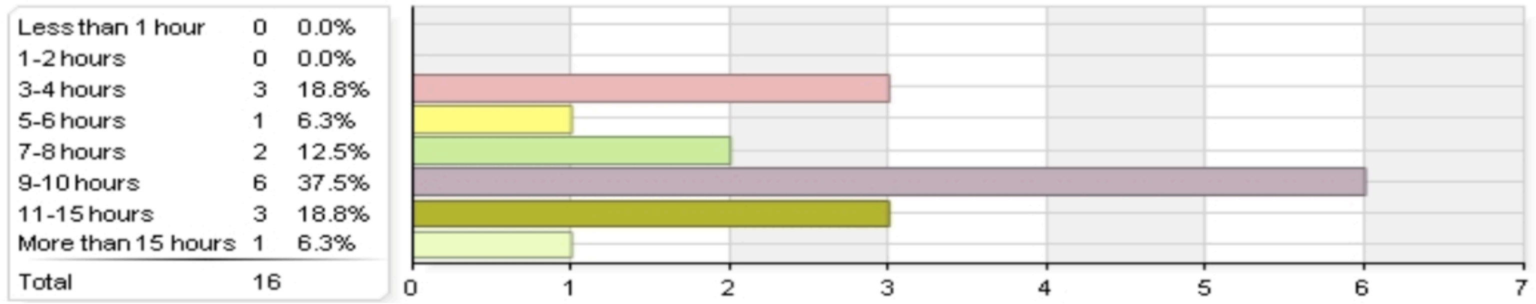
- We expect you to put a lot of effort into the class.
- We expect that you will do this because you enjoy the work.

How effectively did out-of-class work (assignments, readings, practice, etc.) help you learn?



Very Effectively	10	62.5%
Effectively	3	18.8%
Somewhat Effectively	3	18.8%
Not at all Effectively	0	0.0%
Total	16	

In a typical week, about how much time did you devote to this course? (Do not count scheduled class time, labs, etc.)



Less than 1 hour	0	0.0%
1-2 hours	0	0.0%
3-4 hours	3	18.8%
5-6 hours	1	6.3%
7-8 hours	2	12.5%
9-10 hours	6	37.5%
11-15 hours	3	18.8%
More than 15 hours	1	6.3%
Total	16	

Course Structure

- The course will be based on projects.
  - Phase 1 and Phase 2
- Phase 1: The first two (maybe three) projects will be structured in order to create a mockup of Airavata 2.0
- Phase 1 projects are fully working codes, complete with version control, automated installation, integration, and deployment.
- The Phase 1 code base will be used to investigate real questions we have about Airavata’s architecture and deployment.

Phase 2 Projects

- Phase 2 Projects are based on “themes”
- Themes will be used to investigate one or more real Airavata problems. These will be the subject of biweekly projects.
- You can work on different themes throughout the semester.

Grades

- Projects: 80%
  - There will be 8 projects, based on themes
  - These will use GitHub, CI/CD, and Amazon as many of you know.
  - Projects will be due every 2 weeks.
  - Each project will conclude with presentation and report
  - Grading will follow Fall 2016 guidance sheet
  - Grading will be based on individual effort
- Attendance, participation in class, and out of class interactions (via GitHub): 5%
- Final Report: 15%

Theme	Description
APIs, Data Model Representation and Storage	API Comparisons: REST, Swagger, Thrift, ProtoBuff etc.; Optimal serialization, deserialization, storage and search strategies; Internal (CPI) versus externally exposed (API) data models
Distributed Workload Distribution	Load Balancing Stateful vs Stateless Services; Message based vs hybrid message-RPC vs pure RPC; Work-Queue vs Master-Worker load balancing
DevOps Strategies	Fault tolerance testing and validation; Load balancing testing and validation; Continuous upgrades without downtime; Capacity testing; Comparison of hosting alternatives: OpenStack (Jetstream), Amazon AWS, Google, Microsoft Azure
Cyberinfrastructure Integration	Interacting with remote computing resources and data; Metascheduling; Gateway metadata and provenance; Alternative resource utilization
Cybersecurity	PHI, HIPAA, etc alignment for gateways; DevSecOps practices; Auditing, monitoring, and event detection; Client assurance
Data-Centric Computing and Gateways	Airavata and data stream processing; Internet of Things and Airavata; Searching and data mining structured data sets; Event-driven computations
Scientific User Environments	Gateways and interactive computing; Application monitoring; Visualization; Modern user interfaces and environments for science; Notebook integration
Application Toolboxes for Airavata Applications	Defining toolboxes: application and resource collections; Toolbox templates; Packaging, integrating, and sharing tool boxes

Mid-Course Corrections

N.B. This is our first time to do this type of course. We may adjust things as we go.