CSE 451 Project Guidelines

This document describes what you can/must do for the course project for CS 451. The project is meant for you to think about and showcase your skills in building/evaluating a system. The project can be anything really - I have suggested three classes of project along with a few sample projects in each class here.

1 Infrastructure Project

The goal of this project is to implement a significant piece of distributed systems infrastructure. This need not be an original idea or service, as the goal is to get implementation experience. Thus, it is fine to build your own implementation of GFS or DynamoDB. Suggestions

- Implement your own version of map-reduce
- Key-value store with your own choice of consistency/availability/durability metrics
- Kerberos authentication protocol
- Distributed file system
- A load balancing cache, perhaps for a file system
- A cluster-scheduling service
- A file system or key/value store using Paxos or Byzantine fault tolerance
- A lock service like Chubby
- A distributed tuple space for XML documents.

2 Research

You may choose a project topic of your choice. The goal is to perform original research addressing any topic in distributed systems or cloud computing.

Some Ideas

- Perform a performance comparison across different infrastructure implementations, such as predictability, peak performance, average performance etc. This could be for any piece of infrastructure suggested in the previous section. The goal would be to come up with recommendations of a specific implementation based on an empirical study.
- Build a mechanism to seamlessly transfer applications between local infrastructure and cloud infrastructure - bursting as its known in cloud terminology.
- Different instances in a cloud provider may yield different levels of application performance. Can you come up with a scheduling service that chooses where to run your code for maximum performance/minimum latency?
- There are a wide variety of options for replication that balance consistency, performance, availability. Come up with your own protocol for your own needs.
- Security is an often-overlooked issue in distributed systems. Cloud providers have the potential to improve security by automatically encrypting data between hosts or handling SSL for clients. Create a mechanism/protocol to automatically encrypt data to and from cloud tenants.
- Map/reduce has been a rich source of research projects. Take an implementation of map/reduce and extend its functionality or solve a performance problem in it.
- Try injecting faults into a piece network services, such as Cassandra, and see how it responds. Is it really fault tolerant?

3 Applications

Build your own application!

- A DC++ like application for file sharing on Android phones.
- A time synchronisation app for mobile phones.
- Real time chat on mobile phones.
- Distributed image processing either on Android phones or on a LAN platform. Suppose you take a picture and wish to tag it with some set of attributes.

There are endless possibilities here.

3.1 Evaluation

It is challenging to evaluate a new application, but still necessary. You should evaluate these aspects of your application:

- Performance - how much load can it handle (limited by the resources available to you)
- Reliability - how does it handle failures, such as killing an instance
- Correctness - does it actually do what it is supposed to – here you need to define what correct means

4 Milestones

Rather than a big bang evaluation at the end, I would like to impose 2-3 intermediate milestones:

1. Project spec and team: **March 15th**
2. Design (experimental design, software design....): **March 27th**
3. Final presentations: **Week of April 13th in class**

5 Grading

Your project will be graded based on the following rubric:

1. Clear specification (either of the problem or a set of requirements for an application) that are relevant to the distributed systems aspect of the application.
2. A design that addresses each of the distributed systems requirements. These will mostly be protocols and APIs.
3. Completeness of your implementation.
4. A detailed presentation of upto 20+ minutes where you showcase and demo your project.

The grading is necessarily subjective to some extent and all decisions of the judge (namely me) are binding and cannot be contested!