CS441 – Engineering Distributed Objects For Cloud Computing

Course Project

Total: 25% of your final grade

As part of taking CS441, you will work on a course project that allows you to get good hands-on experience with programming distributed objects for cloud computing. A goal of this project is to create and use distributed objects using a variety of different technologies, and these objects should interact with backend relational and so-called NoSQL databases as well as publically available services that can be accessed using the Internet. This project is different from the others that you worked on, since it has no precisely defined specification for an application that you will implement – you will create it! However, it has a clear definition of the direction that you should take and what is expected from you – to show enough complexity of utilizing different technologies for engineering distributed objects, so that the resulting application is nontrivial.

With explosion of social networking and data mining applications that operate on "big data," a marketing term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools[1], a main engineering effort is centered on achieving good performance of these applications. An industry trend, among others, is to use NoSQL databases, which is a buzzword and a misnomer for non-relational data management systems. Essentially, the idea is that the ACID principle is too strong for many applications that use back-end databases, and in many cases, this principle can be relaxed and applications can tolerate occasional inconsistencies. Relational databases are often ACID-compliant, and when many transactions from different processes are sent to a back-end relational database, it becomes a bottleneck, since it enables only limited concurrency. Therefore, many organizations combine relational databases with NoSQL storages to increase the throughput (measured as the number of transactions executed per some unit of time, where a transaction is a collection of requests) at the expense of consistency of data.

Your task is to design an application that processes "big data" using a distributed architecture. Your application should be written in Java (or Scala for additional 5%) and I

[1] http://en.wikipedia.org/wiki/Big_data

ask you to use IntelliJ IDE. Your application will use Akka[2] actors with Spray[3]. You Akka actors will communicate with web services to obtain data in a format of your choice and to store them. Akka dispatchers will invoke filters that will process incoming data and store results that may invoke other filters to obtain more aggregated information. You project will be deployed in Google Cloud for which you should receive your educational coupon – I posted the instructions how to do it. You can download the Google Cloud plugin for IntelliJ[4] and use it to deploy Java backends for your cloud application to the Google App Engine. The actors in your application will interact with the storage using RESTful endpoints, which you will make accessible to clients using the app engine. Your application will log information using SFL4J or some other popular logging tool and you will use Google Stackdriver Logging to analyze your log data. You will use Google data stores for storing some of your data. In general, you will explore the developers' resources at Google Cloud[5] and make use of as many resources for your project as your imagination allows you. Please remember that a goal of this project to get hands-on experience with writing and debugging a distributed application for the cloud, not to follow a precisely scripted set of instructions to obtain a predefined value.

If you need to use different languages for web services of your application, I will accept it as long as you justify your using different languages. You should create and run a number of tests using Junit/Mockito/Scalatest, collect results and compare them with expected values (when possible), so that you make sure that your application behaves as expected. More importantly, you should develop performance tests using Apache JMeter[6] that load-test your application with different numbers of virtual users who request different services at the same time.

Your main job for this course project is to use the Ohloh free, REST-based Application Programming Interface (API) to the Black Duck Open Hub open source directory[7] to analyze software projects in this very large open-source repository. You will request API keys for your application, and the limit is 1,000 requests per day. Since we

[2] http://akka.io/
[3] http://spray.io/
[4] https://cloud.google.com/intellij/
[5] https://cloud.google.com/products/#developer
[6] http://jmeter.apache.org/
[7] https://www.openhub.net/tools

have 50 students in this class and assuming that each will analyze 10,000 applications from OpenHub, the total number of the analyzed applications will be 500,000. I will create a column in the gradebook on the Blackboard where I will specify the starting number of the application range for each student. For example, if you see your starting number 20,000, it means that you can access your applications in the range of [20000, 29999][8].

Since each application has a large number of different attributes (e.g., language, # of commits), you will obtain values of the attributes of your choice using the API calls. You will decide what attributes (and maybe the source code) of the application you will want to use and what information you want to extract and aggregate. For example, would you want to determine what attributes of the applications can predict the frequency of the commits. To do that, you will use RapidMiner or Weka API calls to perform pattern extraction over the collected data. Each of you will decide what interesting questions to ask (see an example here[9]) and to define a rationale for each of these questions. A result of this exercise should be a requirements document that will describe what your application will do with the data. The document may loosely confirm to IEEE/ANSI 830-1998 – you will find many samples of software requirements specification (SRS) documents on the Internet. Please check with me about the chosen format of your SRS. Since you may think of new interesting questions as you develop your application, you will update this document on the ongoing basis.

The second step is once you define the data that your application uses and relationships among these data elements, you will produce different database schemata that describe this data and relationships. When designing your databases, you should figure out how your applications use this data, that is you will revisit step 1, possibly multiple times. There is no need to keep data in your database if your application has no use for this data – if you decide to put some data attributes in your schemata, you must have some methods in your application that use this data. In addition, you should decide what data you put in a relational database and what data goes to a NoSQL database. For

[8] https://www.openhub.net/p/20000
[9] https://www.cs.uic.edu/~drmark/index_htm_files/Treasure.pdf

NoSQL databases, I suggest that you consider Neo4J[10], MongoDB[11], and Riak[12], although I am open to other open source NoSQL databases. I suggest that you will have a web service or a messaging service to communicate with the database.

Each project submission should include at least one back-end relational and one NoSQL databases, together with their logical and physical design. That is, the database schema and SQL statements that realize it should be submitted along with instructions on how to create and populate the database. You should use open-source relational database engines, for example, MySQL or Apache Derby. Your application should retrieve data from and insert and update data in your databases using Structured Query Language (SQL) statements. For NoSQL databases, you should use appropriate API calls that are provided by a given NoSQL database engine of your choice.

You are added as a member of UIC_CS441_2016 team in Bitbucket. Separate repositories will be created for your course projects. You will receive a URI to the git repository for your course project. You will fork this repository and your fork will be private, no one else besides you, the TA and your course instructor will have access to your fork. You can commit and push your code as many times as you want. Your code will not be visible and it should not be visible to other students. When you push it, your instructor and the TA will see a separate private fork. Making your fork public before December 16, 2016 will result in losing your course project grade. For grading, only the latest push timed before the deadline will be considered. If you push after the deadline, your grade for the homework will be zero. For more information about using git and bitbucket specifically, please use this link as the starting point https://confluence.atlassian.com/bitbucket/bitbucket-cloud-documentation-home-221448814.html.

This is an individual course project, so it cannot be done collaboratively. However, I allow you to post questions and replies, statements, comments, discussion, etc. on Piazza. Remember that you cannot share your code and your solutions, but you can ask and advise others using Piazza on where resources and sample programs can be found on the internet, how to resolve dependencies and configuration issues, and how to

[10] http://neo4j.org/
[11] http://www.mongodb.org/
[12] http://wiki.basho.com/Riak.html

design the logic of the algorithm. Yet, your implementation should be your own and you cannot share it. Alternatively, you cannot copy and paste someone else's implementation and put your name on it. Your submissions will be checked for plagiarism. When posting question and answers on Piazza, please select the appropriate folder, i.e., project to ensure that all discussion threads can be easily located.

You will be evaluated based on comprehensiveness of your own design and sophistication of your implementation, the use of cloud technologies, the comprehensiveness of your tests, the number of classes and methods and their sophistication in your application and your documentation. The course project submission date is The course project submission date is Friday, December 9, 2016 at 9:00PM with confirmation email to me drmark@uic.edu, CC to the TA. Failure to submit your project by the deadline shall result in losing your project grade.

Good luck and have fun!