Distributed Enabling Platforms

Final Exam Project
2015/2016

Project Goals

- Projects require students to develop software in Java.
- You must prove you master some concepts discussed during the course, in theory and in practice.
- Some project ideas are proposed by the professor.
- Students can propose their own project ideas, but they must be accepted by the professor.
- Projects are mandatory to successfully pass to the final oral examination.

Project Delivery and Discussion

- A project must be delivered at least two weeks before the oral examination.
- There are no pre-set dates: contact the professor when ready.
- Deliver (via code repo: github, bitbucket, etc) the source code, complete with any data and documents to allow anyone to compile, deploy and test the software.
- Deliver via email an electronically written project report describing the project and the design choices, implementation details, testing procedures and experimental evaluation.
- Every group (1 or 2 students max) must present 10 slides per student during the oral examination discussing the project and its outcomes.
- Be ready to answer queries on code organization, design choices, etc.

Common Requirements

- You must program in Java.
- You must collect and provide input data.
- You must implement testing procedures to debug your code.
- You can use common support libraries (e.g., junit, log4j, etc) that do not overlap with the project goals. Describe them and motivate their use in the project report.
- Do not hard-code configuration parameters, use configuration files/services.
- You must test all the functionalities of your developed system/service and present and discuss the testing results in the project report.

Common Requirements

- Your software must run on a distributed platform (even if developed in pseudo-distributed modes).
- Your software must deal to some degree with autonomous entities contending shared resources.
- Your software must deal to some degree with some form of shared state distributed among nodes.
- Your software must deal to some degree with elasticity, scalability and/or fault tolerance.

Project 1: PAD-FS

- Implement a distributed persistent data store supporting a consistency model you choose.

- Simple API: put, get, list.

- Flat or hierarchical.

- Use the code we discussed during lessons

- A fully-fledged solution like Dynamo is not requested, but gossiping and versioning at least

Project 2: PadBook

- Implement an eventually consistent distributed publish-subscribe system with a social network interface.

- Design the API keeping in mind how it will be exploited.

- Choose the right consistency model!

Project 3: Connected Components in Hadoop

- Implement a software tool to perform connected components identification in large graphs
- Must be a library, focus on simplicity of use for a programmer.
- We will try it on an real Hadoop cluster to test its usability.
- Ask for real-world datasets

Project 3: Connected Components in Hadoop

- Hash-to-Min, in *Finding Connected Components in Map-Reduce in Logarithmic Rounds* by V. Rastogi et al., http://arxiv.org/pdf/1203.5387.pdf
  - Implement the stopping algorithm for an high mark
- LargeStart-SmallStar, in *Connected Components in MapReduce and Beyond* by Kiveris et al., http://dl.acm.org/citation.cfm?id=2670997
  - Implement two versions, with and without the DHT, for an high mark