Project Proposal
Location-Aware Computing in the Hybrid Cloud

1. Vision and Goals Of The Project:

Large datasets usually reside in public cloud provided by third party, such as Google Cloud Services (GCS) and Amazon Web Services (AWS). However, a company may desire certain calculations and information to only reside in private cloud. Moving large amounts of data to private cloud is expensive and time consuming. This project aims to build an application that achieves computational locality with "Function as a Service" (FaaS), and consequently reduces the software development lifecycle. In this way, calculations and queries can be implemented in a public cloud, and then the intermediate result can be transferred back to private cloud. High-level goals of this project include:

- Enabling computational locality via FaaS: This application will transpile the input functions so as to call its equivalent cloud functions in different cloud.
- Reducing development lifecycle: With this project, components in different clouds could be considered and integrated as a single application, therefore reducing the developmental complexity.

Users Of The Project:

This project would be used by companies and research teams that have multiple applications or computational modules in different clouds. Examples include investment banks, hedge funds, or tech companies driven by concerns for safety, privacy, or amount of computation.

2. Scope and Features Of The Project:

- Babel Plugin to transpile functions
  Implement Babel plugin to transpile Node.js functions deployable by FaaS such as AWS Lambda and Google Cloud Functions.
  - Support global functions, lambda expression and class member functions (ES6).
  - Generate a .json file containing all the dependencies.
  - Able to transpile more than one annotated functions.

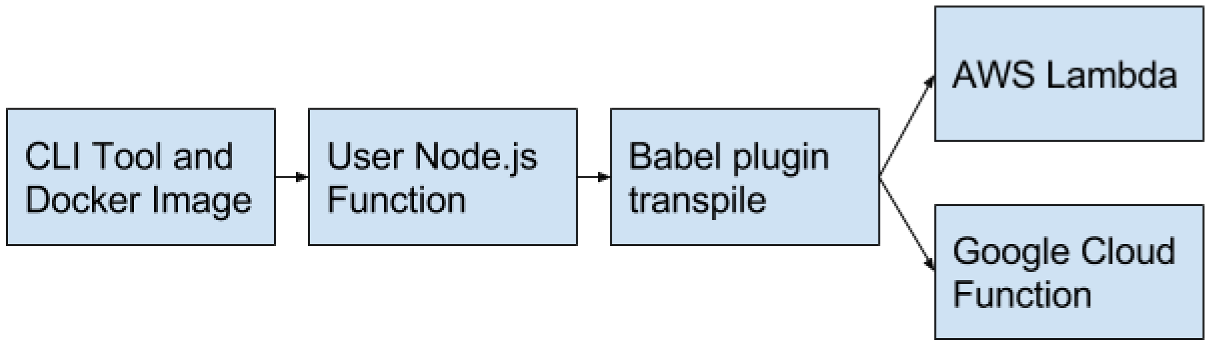
- Deployment Command Line Interface (CLI) tool
  Implement serverless-CLI tool, which generates a docker image with the "local" code and serverless descriptions of the AWS cloud functions
  - Able to deploy the transpiled functions on the AWS Lambda in one command.
  - When redeploying, the CLI tool should only redeploy the functions that have been changed.

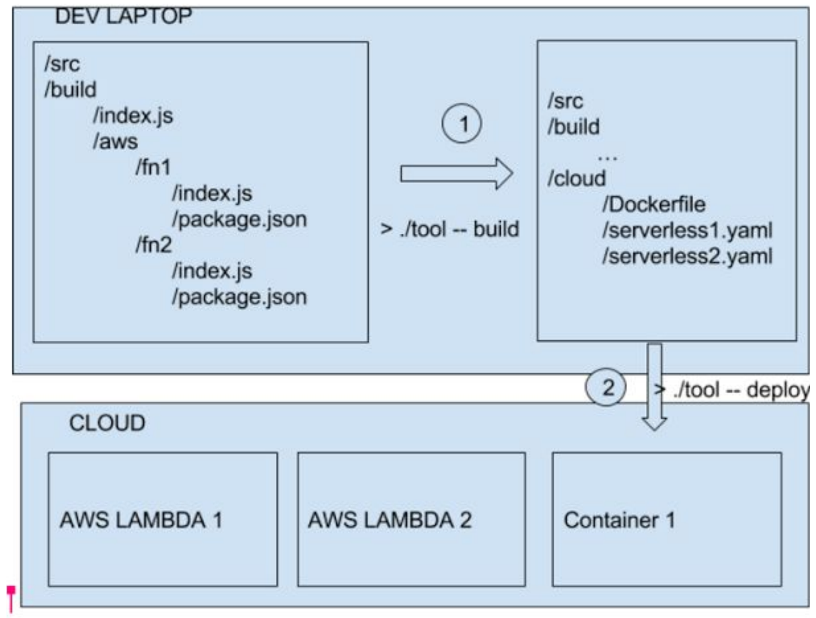
- Sample motivational application
  Develop an application that leverages some higher order functions into different cloud services.

- Bonus: Unify the API from different cloud providers such as AWS S3 and GCS.

3. <u>Solution Concept</u>

Global Architectural Structure Of the Project:





The above figure illustrate the process for FaaS: The CLI tool deploys the docker image with the user Node.js function and AWS configuration on local machines. User function is then transpile through Babel plugin to deployable code on the cloud. Finally, function is deployed on cloud services through the CLI tool.

Design Implications and Discussion:

The following are some key reasons of design decisions and the motivation behind them.

- Functions as a Service (FaaS):
    - FaaS is a new trend in cloud computing where instead of deploying the whole application in containers, only the necessary functions are deployed. Then, these functions listen and react to external events, such as HTTP request or incoming queries.
    - In this project, FaaS servers as the key basis for several reasons:

        - ***It enables the computational locality***: functions and queries could be executed in the cloud where their targeted database lies in, therefore reduce the latency and uncertainty during web communication.
        - ***Node.js support:*** Currently, both Google and Amazon, two of the biggest cloud providers have provided FaaS. Meanwhile, they both support the node.js cloud languages, which has an excellent ecosystem. These build a solid foundation for our project.

- Babel:
    - Babel is a compiler. At a high level, it has 3 stages that it runs code in: parsing, transforming, and code generation. It performs lexical and syntactic analysis in order to build an Abstract Syntax Tree (AST) to represent the code as nested objects. Babel supports various plugins that affect the transformation stage. These plugins stage allows quick and easy transformations to the various Javascript standards.
    - In this project, babel plugin is used to transpile the local functions to the corresponding cloud functions in AWS Lambda or Google Cloud Platform.
- Deployment CLI:
    - Container: The Docker container wraps a set of software in a complete filesystem which provide a homogeneous environment for the machines to run. A automatic generated docker file should be implemented for the ease of deployment.
        - Docker has enabled us to keep consistency among different local machines and containers with predefined configuration files.
    - Service provider: Google Cloud Platform and AWS both provide FaaS. However, Google Cloud Function is still in early stage of development. It seems AWS Lambda is a better option.
- Applications:
  Both GCP and AWS have large amount of users and datasets.
    - The project can be implemented as a application that leverages some higher order service in two different cloud.
    - Or the project can be a RESTful service written as a traditional app but runs as a many cloud function.

4. Acceptance criteria

The minimum acceptance criteria contains two parts:

1) Working babel plugin that generate deployable functions.
    a) Once deployed, the code needs to run without any glitches.
    b) This plugin could be limited to only support class member functions.

2) CLI tool that generates a docker image with the "local" code and deploy the workable functions on different cloud.
    a) It should be able to deploy everything in one command.
    b) During redeploying, the CLI tool should only redeploy the functions that have been changed.

5. <u>Release Planning:</u>

Further detail can be found on Trello:
https://trello.com/b/yDgoTVTa/location-aware-computing-in-the-hybrid-cloud
Releases on GitHub
https://github.com/BU-CS-CE-528-2017/Location-Aware-Hybrid-Cloud

Sprint 1 (due by 23 Feb):

- Babel-plugin with initial support for class member functions.

Sprint 2 (due by 16 March):

- CLI tool to do a simple deploy cloud functions & enhanced plugin support.

Sprint 3 (due by 30 March):

- CLI tool with redeploy functionality & dockerfile generation.

Sprint 4 (due by 13 April):

- Motivational application & enhanced CLI support (cloud fn deployment and dockerfile deployment)

Sprint 5 (due by 27 April):

- TBD - like in any new software project we are going to get stuck here and there at different places or might come up with more functionalities,such as unify API's. This is the time catch up or get creative.