TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Progetti

Corso di Sistemi Distribuiti e Cloud Computing

A.A. 2022/23

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

Project choice and deadline

- Send me an email by February 20th 2023 containing the following info:
  - Subject: SDCC prenotazione progetto
  - Team members (name and email)
  - Chosen project (and description of the application, if applicable)
- Maximum number of available slots for each project
  - Assigned with FIFO discipline: after my presentation ☺
- Communicate promptly and motivate any change to group
- Project is valid only for A.Y. 2022/23

Project delivery

- When to deliver
  - By September 2023
  - About one week before project presentation
  - No prefixed dates for presentation
- What to deliver
  - URL pointing to cloud storage or code repository containing: project code, report, and (if pertaining) datasets of experimental results
  - Write report possibly as a scientific paper (maximum 8 pages using ACM or IEEE format)
    - ACM proceedings templates http://bit.ly/1UbvNLJ
    - IEEE proceedings templates http://bit.ly/1lunzPr

Project presentation

- All team members discuss their project on same day
- What to present (for both project types)
  - Prepare presentation with slides
  - Prepare live demo of project
  - Each team member discusses a part of the project (it is your choice which part)
  - Maximum **10 minutes** per member
    - I will check time and interrupt you if needed
    - Live demo is not included!
  - Questions during and at the end of the presentation

Common requirements for all projects

- Programming language: depends on chosen project
- You can use support libraries and tools to develop your project (of course they should not overlap with the project goals!)
  - Be careful: their use must be properly mentioned in the project report
- System/service with configurable parameters (no hard-coded!)
  - Through a configuration file/service
- You must test all the functionalities of your developed system/service and present and discuss the testing results in the project report

Common requirements for all projects (2)

- System/service state should be distributed
  - The only allowed centralized service can be one that supports service discovery, users logging, and other housekeeping tasks
- System/service supports multiple, autonomous entities which may contend for shared resources
- System/service supports update to some form of shared state
- Depending on chosen project:
  - System/service scalability and elasticity
  - System/service fault tolerance
    - In particular, system/service continues operation even if one of the participant nodes crashes (optionally, it recovers the state of a crashed node so that it can resume operation)

Grant for cloud services

- You can use Amazon Web Services (AWS) through Learner Lab provided by AWS Academy
  - You should have received the email to access the grant, let me know if you cannot find it
  - 100 $, renewable each year
  - Some limits, check the list of available services!
- Plus AWS Free Tier for 12 months (unless you have already registered for AWS account)

Project types

- See first lesson of the course
- Type A
  - Final score:
    - 50% written exam (plus elective oral exam)
    - 50% project (2-3 students per team)
- Type B
  - Final score:
    - 75% written exam (plus elective oral exam)
    - 25% individual project
- Changing the project type (from A to B or viceversa) is not allowed

Projects A overview

- Project A1: Data storage in the cloud-edge continuum
  - 3 students per team, max 6 students
  - Resource constrained devices at edge, more powerful servers in cloud
  - Serve popular/small files at edge
  - Offload storage of unpopular/large files to cloud
  - Cache at edge

Projects A overview

- Project A2: Microservice application for smart cities
  - 2/3 students per team, max 8 students
  - Application in one of the following sectors:
    - Sustainable agriculture and circular economy
    - Improve waste management
    - Innovation and incentive to use public transport
  - One team can participate to CINI Smart Cities University Challenge
  - Deadline: July 15th (July 31st: online meeting with challenge organizers and teams from other universities)

Projects A overview

- Project A3: Serverless in the cloud-edge continuum
  - 2/3 students per team, max 8 students
  - Manage functions in containers
  - Keep containers warm at the edge to avoid cold start
  - Offload functions from edge to cloud serverless service (AWS Lambda) or to an open-source FaaS framework (e.g., OpenWhisk, OpenFaas) running on EC2 instance

Projects A overview

- Project A4: microservice application vs. serverless application
  - 2 students per team, max 6 students
  - Define a simple application of your choice (e.g., image manipulation and classification) including at least 2 microservices/functions
  - Compare performance of the two applications using a load testing tool (e.g., Locust)

Projects B overview

- Project B1: Chord system
  - 1 student per team, max 5 students
  - Go language
  - Support of join/leave is required
  - Replication is optional
  - Deployment using Docker Compose and EC2 instance

Projects B overview

- Project B2: FaaS management
  - 1 student per team, max 5 students
  - Go or Python language
  - Manage single functions written in a given programming language inside a container
  - Offload to Cloud according to a local decision policy
  - Deployment using Docker or another container engine (e.g., Podman, Firecracker) and AWS Lambda for function offloading

Projects B overview

- Project B3: Distributed PageRank
  - 1 student per team, max 5 students
  - PageRank: the famous iterative algorithm by Google founders to rank web pages
  - Scaled PageRank rule

$$R_{i+1}(u) = c \sum_{v \in B_u} \frac{R_i(v)}{N_v} + (1-c)E(u)$$

  - Large link-graphs: need to distribute computation
  - Use MapReduce approach to distributed computation involved in each iteration of the algorithm
  - Map: for each node *v*, calculate rank for each out-link of *v* and propagate to adjacent nodes
  - Reduce: for each node *u*, sum the upcoming ranks and update R_{i+1}
  - Deployment using Docker Compose and EC2 instance