INF-3200 Mandatory assignment 1

Key-value store

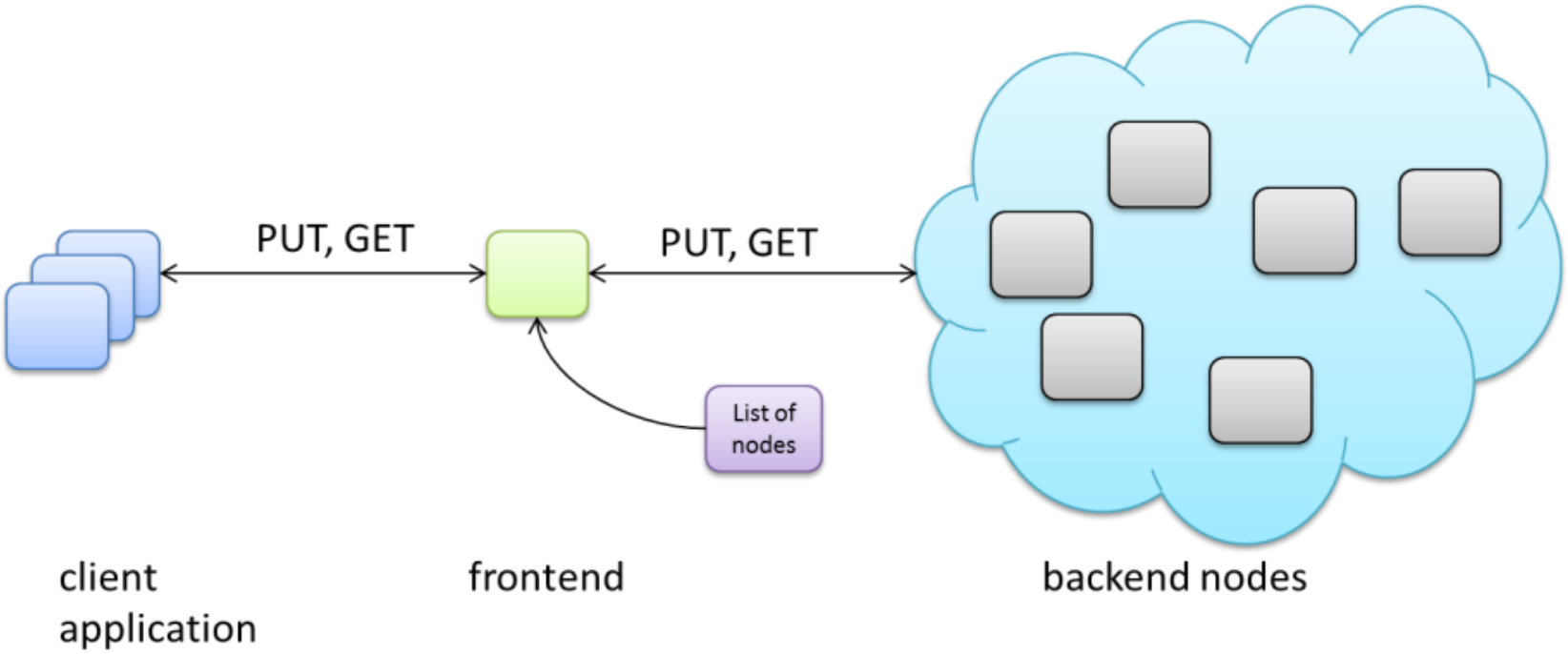
Description

- You are to design and implement a distributed key-value storage system
  - Run on the uvrocks cluster: uvrocks.cs.uit.no

Interface

- PUT
  - Store the message body at the specific URI (key).
  - PUT requests issued with existing keys should overwrite the stored data.
- GET
  - Return the data stored under the URI

System architecture



Distributed data stores

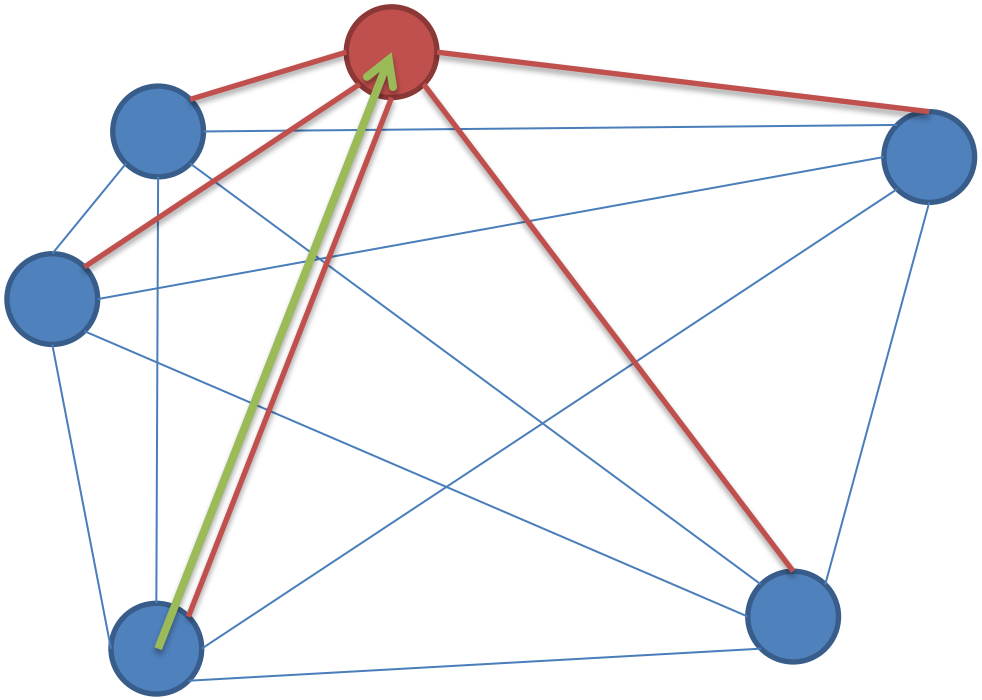
- A network of computers providing a service for storing and retrieving data
- *A key-value store* stores records identified by a unique key.

Peer-to-peer storage systems

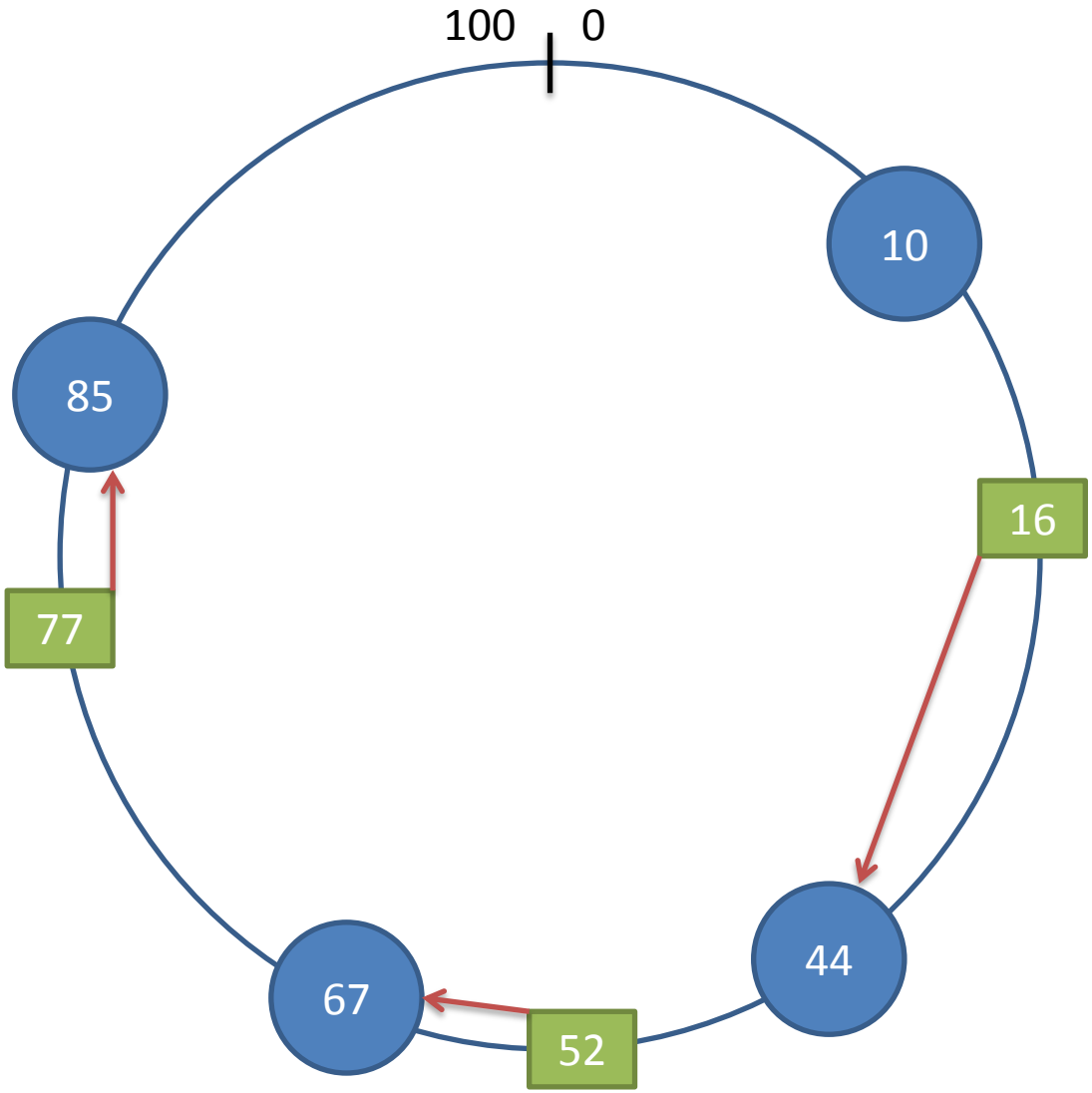
- Suitable architecture for arranging nodes
  - Unstructured and structured
- Distributed hash tables maps a key to one or more nodes
- Example: Dynamo from Amazon

Key location (simple)

- All nodes are queried and the node storing the key responds



Consistent hashing



Backend nodes

- Distribute data storage load between storage nodes according to a chosen architecture.
- Support a minimum of three backend nodes.
- Store the data in-memory.
- Implement a monitoring tool that outputs/visualizes how much data each backend node stores.

Frontend node

- Receives and forwards request to the backend storage nodes
  - You will implement the communication between the frontend and the storage nodes
- Provided on the course github page
- Runs a set of tests to check if PUT and GET operations are working correctly
- The front end must contact a random backend node for each of the request
  - I.e no state is allowed on the frontend except a list of backend nodes

Handin

- Source code (programming language of your choice) with instructions on how to run.
- Report

- You are free to use any language supported by the cluster.
- Groups of two is preferred, but not required.
- On github.
- Deadline is **Monday, October 5th.**

Demo

- There will be a demo presentation in connection with the hand in.
  - You are expected to briefly present your work and demonstrate it.
- We might run our own client applications to check how the system behaves
  - It is difficult to build a “perfect” system w.r.t. fault tolerance and consistency (ref. CAP theorem)

Questions?