Big Data Management
1st Programming Assignment

Lecturer:
Christos Tryfonopoulos

Deadline: Sunday 29/04/2018 until 23.59
Assignment presentation: last lecture

IMPORTANT NOTES:

1. Assignment is done in teams of two. All submitted files MUST contain the names of the team members, their ids (AM), and username/emails in English.
2. After your assignment is complete you must submit it through the Assignments tool of e-class. Only one of the team members should submit the assignment.
3. Assignments sent by email or after the deadline will not be accepted for consideration. Notice that the deadline is firm, and after that the system will not allow you to upload your solution.
4. Please deliver all relevant files (source code, report, dataset, etc.) in a single compressed archive named after the surnames of both team members (e.g., Foka_Grivas.zip).
5. Plagiarism will not be tolerated, and all such cases will be immediately excluded from the course and appropriate action will be taken to notify all involved MSc stakeholders.

Chord-based simulation of a distributed file system

Your task in this assignment is to simulate the operation of a distributed file system implemented using the Chord infrastructure. To do so, you will be given a real-life dataset downloaded from the famous torrent site ThePirateBay and will be asked to create a workload and perform measurements of different parameters.

This project targets to get you acquainted with the actual performance and load-balancing considerations that are faced in a typical distributed file system but without the technicalities often encountered in a real distributed system. For the implementation of the Chord-based simulation you may use the slides from the corresponding lecture (and the references therein). More information for the Chord system may be found in the following seminal paper (will also be uploaded in the related documents of the project):

> I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In SIGCOMM, 2001.

This programming assignment is compulsory and should be implemented in teams of two; the implementation may be done in any operating system and any popular programming language like C/C++, Java, Python, etc. using development tools of your choice. In the final lecture of this semester you have to personally present the solution to one of the examiners; make sure you will have at that time the needed infrastructure to execute your code.

Functionality to simulate

The simulated system will contain N different nodes (N is a system parameter given by the user) and each node will have to store a list of contacts as described in the Chord protocol (successor, predecessor, and fingers). Since you are implementing a simulator, and all nodes will eventually be located in a single machine, communication (i.e., message passing) between the different nodes will be conducted by writing in an appropriate in-memory area that is read by each node. To this end, a node n sends a message to node n' by writing the message to a predefined area of n' that is reserved for new messages. The simulation is carried out in turns, where in the beginning of each turn all nodes read the incoming messages, perform the appropriate actions, and write their outgoing messages to the areas of the recipient nodes.

In your implementation you may consider that nodes do not fail; this means that you do not have to implement the Chord stabilization protocol. Moreover, since you are simulating a distributed system you have to consider that each node knows only the nodes that are present in its finger table (and of course the successor and predecessor node).

The basic functionality that you have to implement for the simulation is:

- the creation of N random IP addresses and ports, that will be used by each node; hashing this information will result in the placement of the node at the appropriate position in the Chord ring,
- the creation of the nodes and the population of their routing information (successor, predecessor, and finger table),
- the assignment of a popularity measure for each file; popularity has to follow a power-law distribution and should be created by you,
- the capability to look up files in the distributed file system based on the Chord protocol.

After you have implemented the aforementioned functionality you have to experiment with different network sizes and different query loads to get measurements regarding the number of messages needed to locate a single file, the observed latency (measured in the size of a chain of messages needed to locate a file), and the load of each node (in terms of requests for a file and messages routed). Please note that in order for your simulation to have meaningful results you need to randomize the node that initiates each query and you also need to average your measurements. Regarding the averaging you have to run the experiments size several times (you have to decide how many) for each network size and select an appropriate data summarization measure (e.g., mean, median, mode or another) for the graph.

Your simulation results should address the following issues for query loads of 100, 1000, and 10000 file requests:

- What happens to the number of messages when the network size (number of nodes) increases? Produce a graph showing the number of messages for networks with 10^2, 10^3, 10^4, 10^5, ... nodes.
- What is the load (in terms of file requests) of each node for a given network size (select the maximum network size that delivers results in a reasonable amount of time).
- What is the load (in terms of routing requests) of each node for a given network size (select the maximum network size that delivers results in a reasonable amount of time).

Comment, analyze and justify your findings appropriately. Apart from the source code and all relevant files, you are also expected to deliver a report discussing:

- detailed installation and execution instructions for your code,
- your experimental setup and design choices that have been made in your implementation, alongside any notable comments about your code (do not include extensive code excerpts),
- the aforementioned graphs along with their analysis and discussion of the results and finings,
- a possible solution (i.e., a load balancing algorithm) to the load (un)balance problem observed in the nodes,
- a short wrap up and summarization section.

Deliverables, grading and bonus

A complete assignment is one that successfully and correctly addresses the aforementioned issues; implementations and reports that cover the requested issues only in part, are graded accordingly. You may also implement extra characteristics (e.g., load balancing functionality, network dynamicity, multi-dimensional data search, etc.) or run extra experiments (e.g., considering other parameters like latency or algorithm variations) are eligible for a 20% bonus.

The assignment deliverables include (but may not be limited to):

- source files (including any necessary libraries)
- executable (if any)
- the data files (subset of the provided dataset) that you used in your experiments along the popularity distributions for each file
- a written report as explained above.