Bilkent University

Department of Computer Engineering

# RENDT

# Remote Execution and Distribution of Tasks

**Senior Design Project**

## Project Specifications

Project Group Members:
Huseyn Allahyarov
Mahammad Shirinov
Ibrahim Mammadov
Cenk Er
Nurlan Farzaliyev

Supervisor: İbrahim Körpeoğlu
Jury Members: Hamdi Dibeklioğlu and Özcan Öztürk

Project Specifications
October 14, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2

## Contents

# Project Specifications

*Project short-name:* **Rendt**

## 1 Introduction

With the ever-growing advancements in computer algorithms, machine learning tools, and with the availability and accessibility of such tools, distributed computing and cloud computing systems have become extremely widespread, to the point of almost being a necessity. [1][2]

The biggest players in these fields are currently Amazon (AWS), Microsoft (Azure) and Google (Google Cloud). What's common among these providers is that they all have large centralized networks of nodes somewhere in a server farm (or several farms), and they provide users with computing power and other services such as storage using parts of that network.

The fact that all the nodes belong to one party and are hosted in one or a few dedicated locations gives them great reliability and ability to offer good pricing. However, since there are few providers and millions of users, only the few big players make revenue. Also, the amount of computing power that is available is dictated by the sole provider.

With Rendt we propose an alternative solution to this vast need of distributed and cloud computing power, where multiple parties can offer their machines and get paid in return, while users will still have access to the computing power that they need for their projects and experiments. Similar solutions have been used in the academic community to solve problems like the decomposition of natural numbers as a sum of three cubes, [3] but they are voluntary in nature and have no commercial use. [4]

## 1.1 Description

Even with all this need for computing power, there is still a large amount of idle or underused machines, which are mostly computers that belong to personal users or small companies that don't have distributed computing infrastructure. Rendt is a platform where these idle resources can be shared with people who need them and turned into profit. Users can choose a resource on their machine that they want to share, choose a type of task they are willing to share it for, and possibly state their price. Then, other users in need of that particular resource will see different offers for their query, choose one they like and start using resources. After the tasks are run and results sent back to the task issuer, the payment will be made and transaction will be completed.

Using Rendt, somebody with powerful GPUs in their laptop who may not need them for personal use, at least not all the time, can turn it into profit, and dually, somebody who needs GPUs to run get some work done (train neural networks) but doesn't have access to any can look for people offering their resources and have access to them without owning any GPUs. Rendt is going to play the role of a match-maker and regulator in such transactions. It will host users that share their resources, along with detailed information about their computing power, types and availability of resources and other information, for example, the time the person is willing to lend their resource or the history of the tasks they have successfully (or unsuccessfully) completed.

Rendt will be used for remotely running different kinds of tasks, such as training neural networks, rendering videos, or plain mathematical (CPU-heavy) calculations. This compartmentalization will help users better find suitable offers, since machines best suited for a particular kind of job will be tagged by that category by the host themselves. Additionally, Rendt will support distributed computations as well: if the user's task is parallelizable, they will be able to request more than one node for their task and run it on them and then receive the single result back, as if running their task on some centralized cluster.

When a user issues a task and selects a node(s) to run it on (or leaves the selection to the system), the task goes to our central server, where a match is registered, and the transfer of the needed files (data, binaries, scripts) is initiated (e.g. by passing the files from issuer to the central server and from there to the host, by P2P file transfer etc.). At this stage, a part (or the whole) of the payment the issuer has agreed to pay is deducted from their account and held by Rendt. The host node(s) start running the task(s), and once done, transfer the result files back to the issuer. Then, the host receives the payment and the transaction is over.

## 1.2 Constraints

### 1.2.1 Implementation Constraints

- Cross-platform supports should be provided by the system to avoid incompatibility issues.
- A secure online payment system should be implemented in order to make payments.

### 1.2.2 Economic Constraints

- Some server maintenance costs should be considered to keep the system running.
- Some APIs to be used may require fees.
- Prices for databases that will store the input and output files and the intermediary files should be considered.

### 1.2.3 Performance Constraints

- P2P architecture puts a fundamental limit on the performance of the system in the distributed mode, because of its remote and disjoint structure. In contrast with the centralized online systems such as AWS or Google Cloud, the communication between the nodes must be done over the internet, since the nodes are not located in the same place and not connected by wires. [5]

- Execution time depends heavily on the connection speeds of the host and the issuing system. Thus, the time it takes to finish a job will be increased with the addition of upload and download times.

#### 1.2.4 Reliability Constraints

- Reliability depends heavily on the users hosting their resources. Connection loss or power outage must be avoided by the hosts, and will be regarded as their responsibility in the transaction.
- Tasks should be executed independent of the executing system's OS, environment or any other specification.

#### 1.2.5 Environmental Constraints

- There will be a need for a server to keep the system running, especially, for database and hosting. Other than that, there is no need for any other hardware usage, instead, project intends to use the existing power of the purchased personal PCs to leverage underused hardware.

#### 1.2.6 Commercial Constraints

- The user base of the project are the people who own the necessary software (e.g. Adobe Premiere rendering software) to help others in need via renting the power of their hardware (e.g. GPUs). However, ideally, for some use cases, there won't be a need for any third-party software.

## 1.3 Professional and Ethical Issues

#### 1.3.1 Privacy

- Our system should work in a way so as not to compromise intellectual or any other private information involved in the process of remote processing. Privacy of information must be established, and our users and their data must be safe.
- Users' personal data stored on our system will not be disclosed without consent, and every individual's privacy using our system will

be respected. We will comply with the general data protection regulation. [6]

### 1.3.2 Payments

- In Rendt, the user who receives data for processing will be paid for sharing his/her hardware. Until the process is executed and the user who sends the task is satisfied and accepts the payment to be delivered, payment will not be delivered to the user who shares their hardware.

# 2 Requirements

## 2.1 User Friendliness

Our system should be user friendly, since its target audience is not only the technically skilled users, but everyone who has a powerful computer or wants to use one. The user interface should be simple and attractive at the same time. All network requests, compilations and executions, data sending and receiving should be done in the background so that the users don’t have to deal with all the technical details. The only thing users should be responsible for is the knowledge about tasks they are sending and results are they expecting for.

## 2.2 Safety

Our system should be extremely safe as the users who interact with each other will not necessarily know each other. The users shouldn’t have the privilege to access each other’s data and control as the only thing there are doing is requesting to send their data which is going to execute on another computer automatically after the second user accepts the request of the first user. All other issues are going to be handled automatically in the backend of our system so that the users don’t have direct access to each other resources.

## 2.3 Privacy

The user who is receiving the task with its files for execution (code, video, image, etc.) should not be able to access or view them. The only thing the receiving user would do is accept first users request and all the job is going to be automatically and seamlessly run in the background without the second user's interference.

## 2.4 Performance

Performance is one of the most important requirements of our project. The scheduling of tasks and requests incoming to the server, the distribution of data through the network channels and the communication of the network packages is extremely important for our project's performance. The system must be designed in a way to maximally optimize scheduling and data flow.

## 2.5 Scalability

The platform should be scalable to be able to host, deliver and maintain large quantities of tasks. This is an important requirement since Rendt is aimed for everyday use.

## 2.6 Reliability

Reliability of the system is very important but as it was mentioned above it depends entirely on users (hosts) and their environments. To handle this issue the system should keep a record of users' reliability (past tasks, completed or ended abruptly). For this the system will generate reliability points so that the requesting users know how reliable different hosts are.

# 3 References

[1] “What is distributed computing and what’s driving its adoption?”. Packt. https://hub.packtpub.com/what-is-distributed-computing-and-whats-driving-its-adoption/ (accessed October 14, 2019).

[2] "The Distributed Computing Paradigms: P2P, Grid, Cluster, Cloud, and Jungle". Dr. Brijender Kahanwal, Dr. T. P. Singh. International Journal of Latest Research in Science and Technology, Vol. 1, No. 2, pp. 183-187, 2012

[3] University of Bristol. "Sum of three cubes for 42 finally solved -- using real life planetary computer." ScienceDaily. www.sciencedaily.com/releases/2019/09/190906134011.htm (accessed October 14, 2019).

[4] “Seven Ways to Donate Your Computer's Unused Processing Power”. Vice. https://www.vice.com/en_us/article/bmj9jv/7-ways-to-donate-your-computers-unused-processing-power (accessed October 14, 2019).

[5] “Comparison Between Wired network & Wireless Network”, Free Computer Networking Preparation https://computernetworkingclass.blogspot.com/2016/08/comparison-between-wired-network-and.html (accessed October 14, 2019).

[6] “General Data Protection Regulation". https://gdpr-info.eu/ (accessed October 14, 2019).