BTP Phase 1

Effective Replica Management in Car Navigation System using Edge Cloud Environment

Lavish Gulati - 170101082
Vakul Gupta - 170101076

1 - Abstract

- ➔ Car navigation system is nowadays based on **centralized** cloud architectures.
- ➔ However, **edge cloud computing** provides robust computational and storage resources improving response times, system scalability and data reliability.
- ➔ The **multi-replica strategy** used in edge cloud computing architecture can create multiple data replicas and store them in different edge nodes, which improves data availability and data service quality.
- ➔ Due to time-varying user demand, the number of data replicas need to be **dynamically adjusted**.

- ➔ Load balancing is also required while **placing** the newly added replicas.

- ➔ We also consider the problem of data replica **synchronization** and data **recovery** in case of failures.

- ➔ We propose an **optimization problem** based on the performance of the edge cloud computing architecture in car navigation system, and improve upon the existing replication algorithms.

2 - Introduction

Car Navigation System

- ➔ A **car navigation system** uses a satellite navigation device to collect various information such as origin-destination stations, occupancy of vehicles, pedestrian activity trends, and more.

- ➔ The on fly traffic information collected from various such navigation systems can be used to plan the **optimal path** to some destination and reduce traffic congestion.

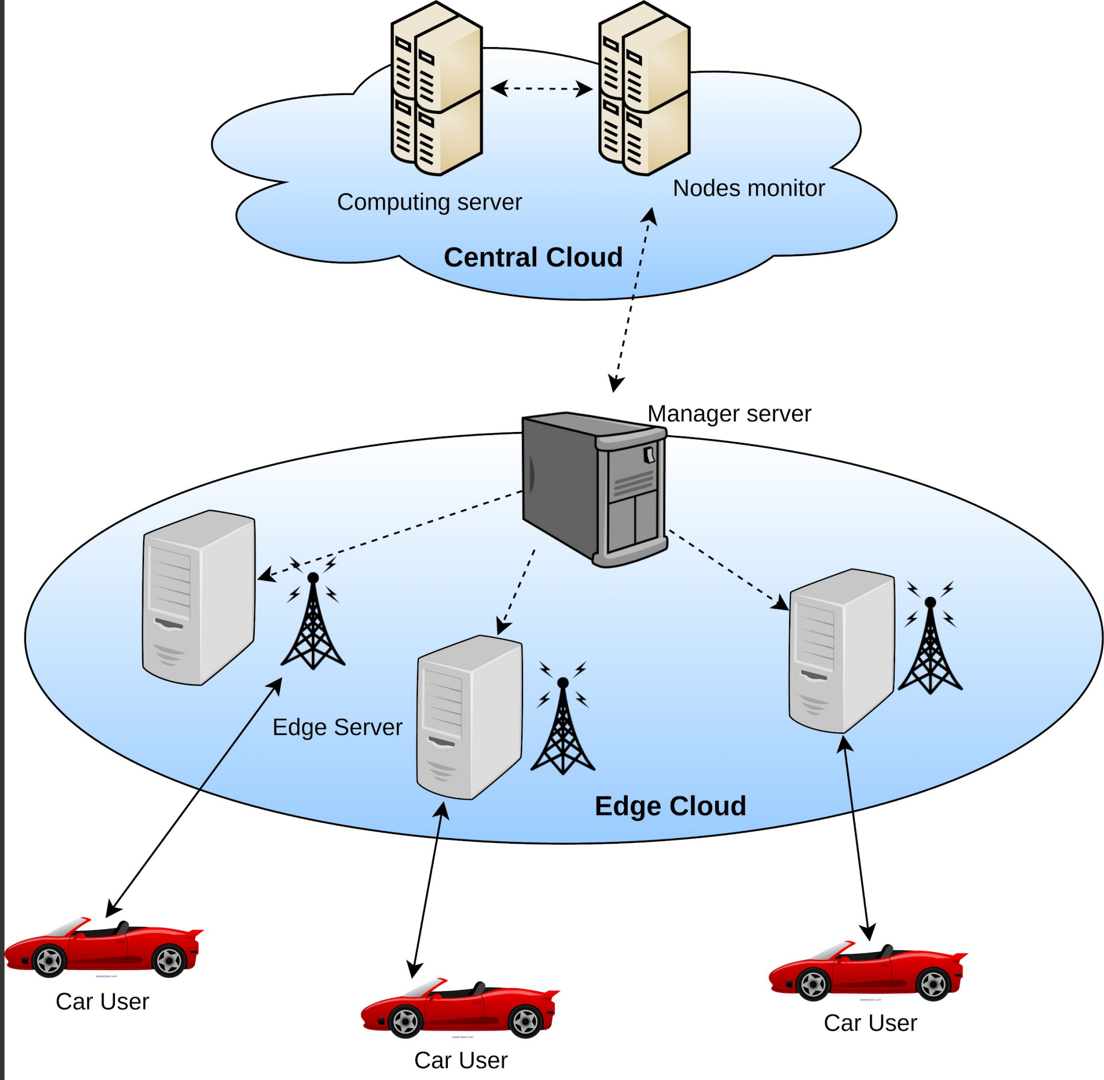
Edge Cloud Computing

➔ **Edge computing** is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed (in our case the car users), to improve response times and save bandwidth.

➔ Since the computing nodes are located over a large geographical region, failures or outages might interrupt the service to the clients which can be solved by **data replication** approach.

➔ **Data replication** is the process in which the data is copied at multiple locations (different edge servers) to improve data availability and data service quality.

Navigation system in edge cloud architecture



Data Replication

➔ The number of data replicas needs to be **dynamically adjusted** because of the time-varying data and computation requests.

➔ The continuously changing number of replicas leads to the requirement of **placing** the newly added replicas on edge nodes.

➔ We also need to ensure **data synchronization** between two or more edge nodes and update changes automatically between them to maintain consistency.

➔ To protect volume data from unrecoverable failure, we need to replicate a volume to a replica node. We can **recover** the data from the replica node once the issue has been rectified.

3 - Motivation

Speed and Security

Speed

Since edge computing devices collect and process data locally in nearby edge nodes, there is no need for the data to be routed to distant centralized servers, thus reducing network latency and response times, and increasing performance of the network.

Security

On the other hand, edge computing distributes applications, storage, and processing over a large geographical region and wide range of computing nodes, which makes it secure against such attacks and failures.

Scalability and Reliability

Scalability

Small edge computing nodes are distributed and located nearer to the clients as compared to large centralized data centers of traditional cloud architectures, and hence the same computational and storage capabilities can be scaled more effectively.

Reliability

Since edge computing nodes are located closer to the clients, a network failure in a distant location will not affect the performance in other areas. Even in case of a data center failure, edge nodes will continue to operate effectively because they process vital functions natively.

4 - Related Work

4.1 - Dynamic replica creation strategy

- ➔ The proposed replica creation strategy takes into account two factors: dynamic and static.
- ➔ The **dynamic factor** is calculated based on data block heat and data access response time.
- ➔ The **static factor** is calculated based on data block size and file size.

$$f_{d,t} = \sum_{i=1}^{n} nc_{d,t}^{i}$$	**Access frequency of data block d** where $nc^{i}_{d,t}$ is number of access for the i^{th} replica of data block d
$$h_{d,t} = \frac{\sum_{k=1}^{n} f_{d,t}(k)}{n}$$	**Average access frequency** where $f_{d,t}(k)$ denotes the access frequency value of the k^{th} time interval at time period t
$$heat_{d,t} = h_{d,t} + h_{d,t-1} + ph_{d,t+1}$$	**Heat in time period t** where $ph_{d,t+1}$ denotes the predicted value of the access frequency at time period t+1
$$ART_{d,t} = \frac{\sum_{k=1}^{n} RT_{d,t}(k)}{n}$$	**Average response time** where $RT_{d,t}(k)$ indicates access response time of the k^{th} time interval at time period t
$$DF_{d,t} = heat_{d,t} \times ART_{d,t}$$	**Dynamic factor $DF_{d,t}$**

$$Q_d = \begin{cases} \lceil F_d / S_0 \rceil & , F_d \% S_0 = 0 \\ \lceil F_d / S_0 \rceil + 1 & , F_d \% S_0 \neq 0 \end{cases}$$

Static factor Q_d

where F_d is the file size of the data block d and S_0 is the file system fragment data block size.

$$H_{d,0} = 0$$

$$H_{d,t} = [\alpha \times H_{d,t-1}(f_j)] + [(1-\alpha) \times DF_{d,t} \times Q_d^{-1}]$$

Optimal Number of Replicas $H_{d,t}$

Here, α denotes the impact factor for data block heat, and is given by

$$\alpha = \Delta T_t / (\Delta T_t + \Delta T_{t-1})$$

where ΔT_t is the time difference from the starting time to the time period t.

4.2 - Replica placement strategy

- ➔ The replica placement strategy is used when the number of replicas of the data block needs to be increased.
- ➔ In that scenario, while ensuring the load balancing of the file system, we need to place the newly added replicas on appropriate edge nodes.
- ➔ To judge the performance, a multi-objective replica placement problem is formulated which is affected by the following 3 parameters, i.e., the file system cluster storage load, the edge node’s performance, and the network distance.

Edge Node Load Performance

$C_{j,cpu} = fr_j \times nc_j \times (1 - uf_j)$	**CPU processing capacity** where fr_j denotes frequency of CPU of the node j, nc_j denotes number of CPU cores, and uf_j denotes usage of CPU
$C_{j,w/r} = S_{j,r} \times a + S_{j,w} \times (1 - a)$	**Disk read/write performance** where $S_{j,r}$ denotes disk read speed, $S_{j,w}$ denotes disk write speed, and $\alpha \in (0, 1)$ denotes the weight parameter
$C_{j,mem} = ms_j \times (1 - mc_j)$	**Load capacity of memory** where ms_j denotes size of the memory, and mc_j denotes memory usage
$C_{j,ds} = D_{j,\mathrm{size}} - D_{j,usage}$	**Load capacity of disk space** where $D_{j,size}$ denotes size of the disk, and $D_{j,usage}$ denotes used capacity of the disk
$\max \quad s_1 = C_{j,cpu} + C_{j,w/r} + C_{j,mem} + C_{j,ds}$	**Sub-objective function** for performance of edge node j

File system cluster storage load

$$\beta_j = \frac{F_j \times B_{\text{size}}}{SS_j}$$	**Usage of storage space of edge node j** where SS_j denotes total size of storage space for edge node j, B_{size} denotes uniform size of data block in the file system and F_j denotes number of data blocks stored in edge node j
$$\max \quad s_2 = \left(\frac{\sum_{j=1}^{N^{\text{node}}} (\beta_j - \bar{\beta})}{N^{\text{node}}} \right)^{-1}$$	**Sub-objective function of file system cluster storage load** where N_{node} denotes the number of edge nodes

Network Distance

$$D_j = \frac{nd_j}{nd_{\max}}$$	**Network distance coefficient** where nd_{max} denotes maximum distance between pair of nodes and nd_j denotes distance between the source node and the target node j
$$\max \quad s_3 = (D_j \times S_0 \times C_{tr})^{-1}$$	**Network distance sub-objective function between edge nodes** where S_0 denotes data block size, C_{tr} denotes transmission cost per bit of data per unit time

Multi-objective optimized replica placement problem

$$\max S(d) = [s_1(d), s_2(d), s_3(d)] \quad \text{s.t. } d \in X$$	**Multi-objective optimization problem**

4.3 - Replica synchronization strategy based on replica delayed update

- In an edge cloud architecture, the edge nodes may be spread over a **large geographical region**.
- In such scenarios, network congestion and insufficient bandwidth may cause **synchronization delays**.
- This will eventually cause **failure** in synchronization of the data replicas, resulting in low write throughput and higher write latency.
- So our replica synchronization strategy should be adaptive to delays because of the heterogeneity of the network.

Selective data replica synchronization

- Suppose there occurs a data block write request from a client and the number of redundant replicas of the requested data block d is N .

- W replicas are synchronized before the success indicator is returned to the client. Here, W is expressed as follows

$$W = (N/2) + 1$$

- This strategy selects W edge nodes holding the replica of the requested data block which are closest to the client in terms of **network distance**.

- These W edge nodes are synchronized, thus forming a data transmission channel for replica synchronization.

Replica status table

- ➔ For the unsynchronized replicas, we define the RST which stores the synchronization status of the data block replicas.

- ➔ Each RST has two entries: RST.Location and RST.Old.
  - ◆ **RST.Location** depicts the edge node that stores the data block replica
  - ◆ **RST.Old** depicts whether the replica has been updated.

- ➔ RST.Old = 1 means that the replica has not been synchronized, and RST.Old = 0 means that the replica has been synchronized.

RST of data block A	
Location	Old
Edge Node 1	0
Edge Node 2	0
Edge Node 3	1

Delay-adaptive synchronization

- The synchronization updates of the N - W unsynchronized replicas will be triggered in the following two cases:
  - The data block that has not been synchronized becomes the hot data.
  - The load performance of the edge node where the replica has not been synchronized is strong

$hah = \frac{\sum_{k=1}^{N} h_{k,t}}{N}$	**Average access frequency at time period t** where $h_{k,t}$ denotes the access frequency of data block k at time period t If $h_{k,t}$ > hah, then data block A is defined as hot.
$alc = \frac{\sum_{i=1}^{J} C_{i,t}}{J}$	**Average load capacity of all edge nodes at time period t** where $C_{i,t}$ denotes the load capacity of the edge node i at time period t If $C_{i,t}$ > alc, then the load capacity of edge node i is defined as strong.

4.4 - Replica recovery strategy based on load-balancing

Failure edge node detection

- ➔ In HDFS, an edge node periodically generates **heartbeat packets** according to its node status and storage status, and sends the heartbeat packets to a specialized edge node denoted by **CollectorNode** at a certain interval, so as to let CollectorNode know its running state.
- ➔ In case of **node failure**, it would not be able to generate the heartbeat packets.
- ➔ As a result, the CollectorNode would not receive the heartbeat packet from the edge node within a certain period of time.
- ➔ The CollectorNode would thus **establish the failure** of the corresponding edge node.

Selection of data blocks to be recovered

$$ap_{d,t} = \frac{co_{d,t}}{sco_t}$$	**Access probability of data block d at time period t** where $co_{d,t}$ denotes count of access of data block d and sco_t denotes the total count of accesses for all data blocks
$$\mathrm{cost}_d = \frac{\mathrm{size}_d + \mathrm{Tseek}_d}{BW_d}$$	**Block recovery costs of data block d** where size_d depicts data block size, Tseek_d depicts time required to locate other replicas of the data block d and BW_d represents maximum available network bandwidth
$$\mathrm{Sel}_d = \frac{ap_{d,t}}{\mathrm{cost}_d}$$	**Replica selection factor for the data block d**
$$ASel = \frac{\sum_{n=1}^{N} Sel_n}{N}$$	**Average value of replica selection factor for all data block in the failed edge node** If Sel_d > ASel, then data block d is chosen as the data block that needs to be recovered.

Selection of the target node

$$P_{\text{disk}} = 1 - U_{\text{disk}}$$	**Disk space availability of edge node** where U_{disk} denotes the using rate for the node disk.
$$P_{CPU} = f_{CPU} \times C_{cpu} \times (1 - U_{cpu}) \times V_{cpu}$$	**CPU load capacity of the edge node** where f_{CPU} denotes frequency of the CPU, C_{cpu} denotes number of cores of the CPU, $(1 - U_{cpu})$ denotes remaining usage rate, V_{cpu} denotes bus speed.
$$P_{\text{cache}} = \frac{L_{\text{cache}}}{S_{\text{cache}}}$$	**Capacity of cache** where L_{cache} denotes cache remaining capacity and S_{cache} denotes time required to access data.
$$CP = (A_1 \times P_{disk}) + (A_2 \times P_{CPU}) + (A_3 \times P_{\text{cache}})$$	**Overall load capacity of the edge node** where the A_1, A_2 and A_3 are the weight coefficients. For recovery of data block d, the edge node with largest CP value not containing the data block d is chosen to be the optimal target edge node.

5 - Objective

To study the performance of edge cloud computing architecture using effective replica management in car navigation system based on different parameters like response time and load balancing capabilities, and in the process maintaining data consistency using synchronization and the capability of the system to recover from failures.

6 - Problem Formulation

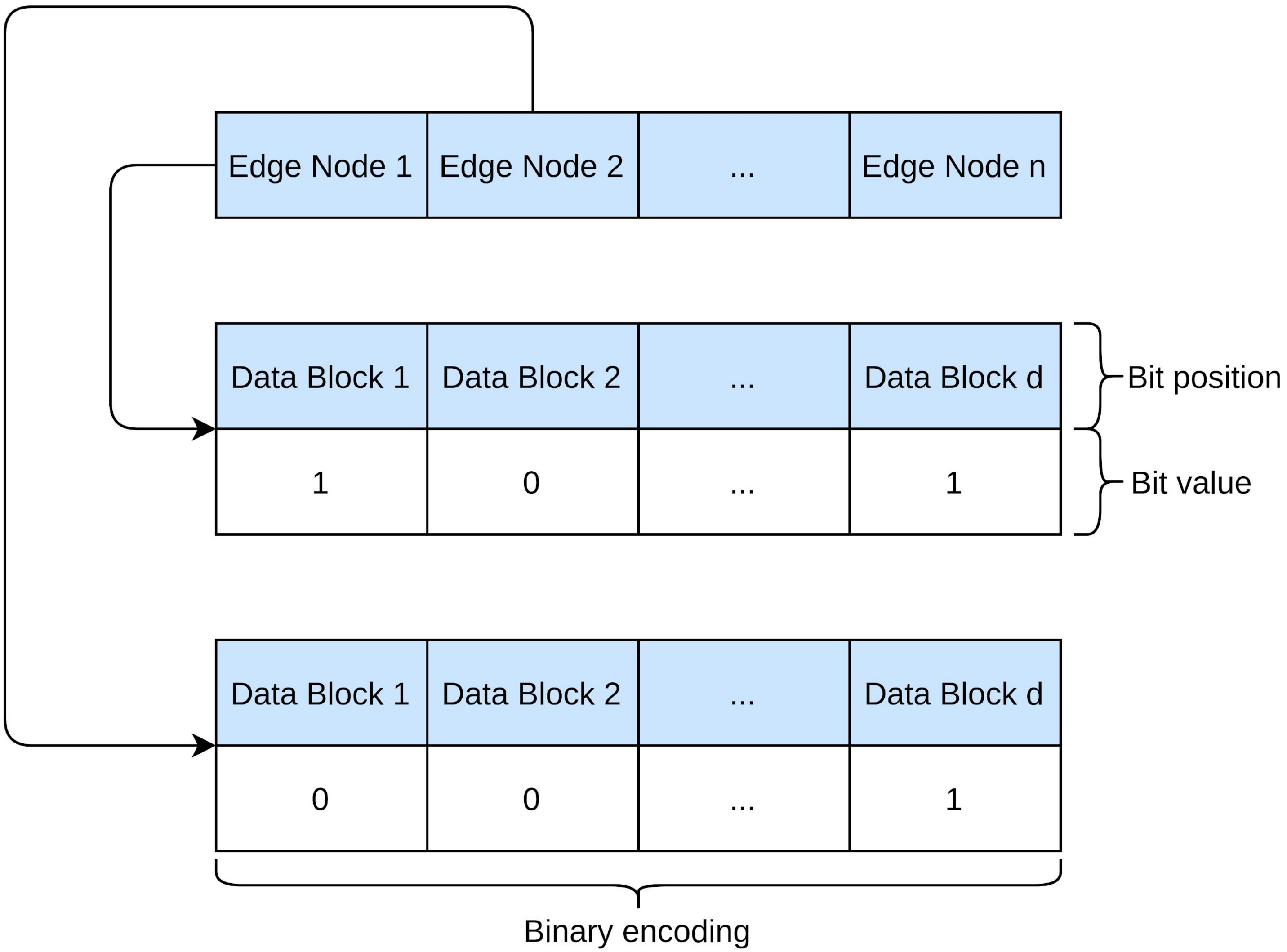
- ➔ Suppose we have a edge cloud architecture A with m central nodes given by $\{C_1, C_2, ..., C_m\}$ and n edge nodes given by $\{E_1, E_2, ..., E_n\}$.

- ➔ The data of each locality is considered as a single data block. Thus, we get a set of d data blocks containing the entire information for all the localities denoted by $(D_1, D_2, ..., D_d)$, where D_i represents the data block of the i^{th} locality.

- ➔ The dynamic replica creation strategy finds the optimal number of replicas $H_{i,t}$ of the i^{th} data block at time period t.

- For replica placement, we assign a binary encoding B_j to each edge node E_j whose length is same as the number of data blocks d and the i^{th} bit in the encoding takes a value of 1 or 0, which means whether a replica of i^{th} data block is stored in the edge node or not.

- The set of n such encodings for all the edge nodes determine where the replicas are placed in the edge cloud system.


Edge Node 1
Edge Node 2
...
Edge Node n
Data Block 1
Data Block 2
...
Data Block d
Bit position
1
0
...
1
Bit value
Data Block 1
Data Block 2
...
Data Block d
0
0
...
1
Binary encoding

The encodings should optimize the multi-objective replica placement performance indicator function S(d) under the constraint that the optimal number of replicas of i^{th} data block is $H_{i,t}$ which can be modelled as

$$\sum_{j=1}^{n} B_{j,1} = H_{1,t}$$

$$\vdots$$

$$\sum_{j=1}^{n} B_{j,i} = H_{i,t}$$

$$\vdots$$

$$\sum_{j=1}^{n} B_{j,d} = H_{d,t}$$

where $B_{j,i}$ represents the i^{th} bit of the binary encoding B_j.

7 - Proposed Direction

- ➔ We will use **Docker containers** to simulate each application interface or computing node. We will use **socket programming in Python** for inter-container communication.

- ➔ The car users send optimal path queries to the load balancer which re-routes them to edge node containers with information about the specified locality.

- ➔ The edge node containers solve the query and return the optimal path to the car users.

- ➔ The manager node will collect information about the processing load, file system load and response time from all the edge node containers and send it to the centralized server.

- ➔ The centralized server algorithmically adjusts the placement of replicas in the architecture to improve performance.

- ➔ The performance of synchronization is measured by (i) the time that it takes to write the data on W synchronized replicas and (ii) the time that it takes to fetch the updated data from $n - W$ unsynchronized replicas.

- ➔ For measuring the performance of data recovery, we need to introduce failures in the architecture.

- ➔ For this, we model a probability distribution for each edge node over a certain period of time. The probability $p_{j,t}$ denotes the probability of failure of edge node j at time period t.

- ➔ We will measure the performance by the average time taken for a failed edge node to recover, i.e. to copy all the data blocks in the edge node to the target node.

Thank you!