Fed-DIC: Diagonally Interleaved Coding in a Federated Cloud Environment

Giannis Tzouros
Department of Informatics
Athens University of Economics and Business

Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business

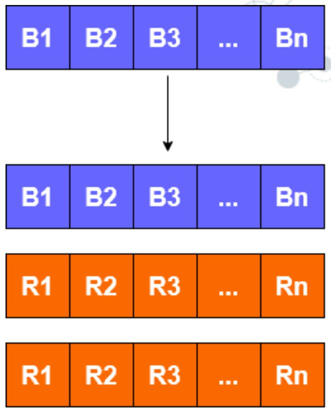
Introduction

- In recent years, the management of big data has become a vital challenge in distributed storage systems
- Failures, outages and unreliable equipment may lead to data loss and slowdowns
- To guarantee availability, distributed systems deploy **fault tolerance** methods

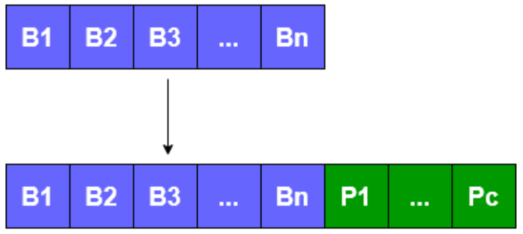
Fault Tolerance Methods

- **Replication**
  - + Simplest form of redundancy
  - + Replicates data content into multiple replicas for data recovery
  - - Massive storage overhead

- **Erasure Coding**
  - + Equal or higher redundancy that Replication
  - + Creates parity data that recover multiple chunks within a data block
  - + Higher storage efficiency
  - - Limited reliability (depending on the # of parity data)
  - - High read and network access cost during repairing processes due to sparsely stored data
  - - The sparsity problem can be dealt by using metadata, but it depends on where the metadata will be stored



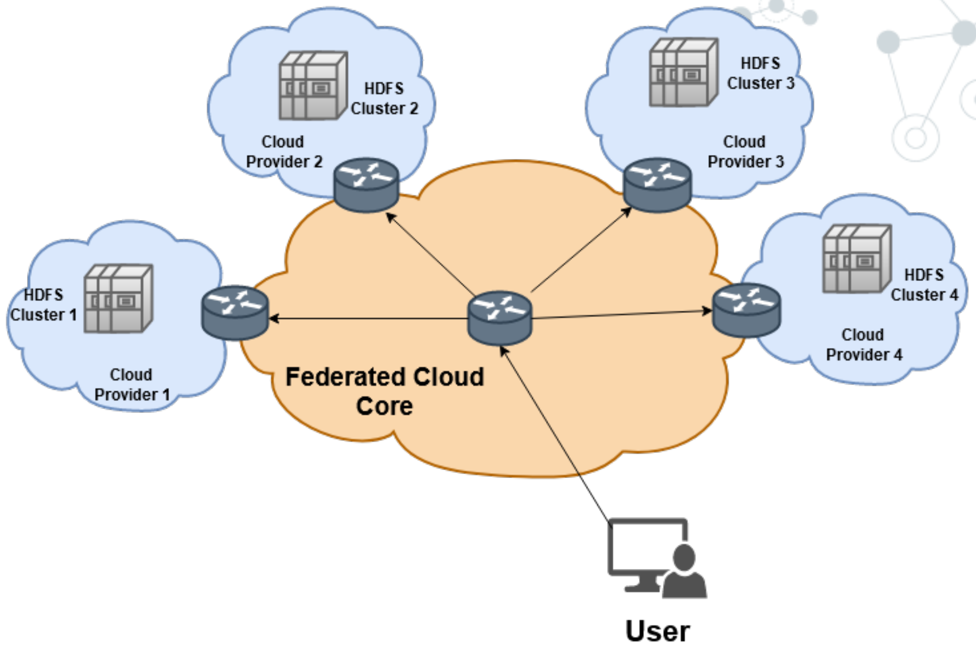
Replication



Erasure Code

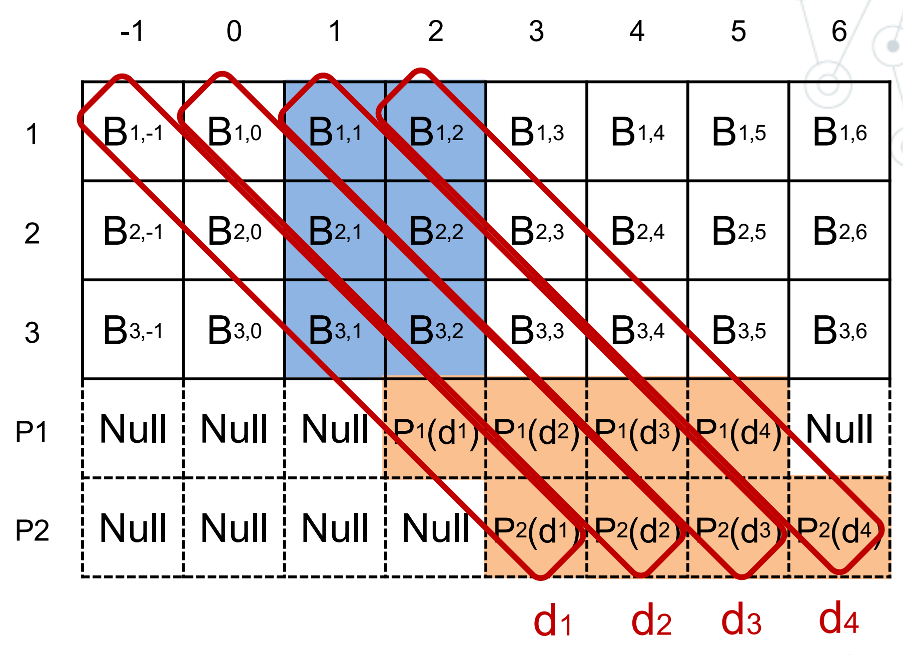
Federated Cloud

- Most popular distributed systems today (HDFS, Azure, Google FileSystem, Ceph) store data into multiple nodes, organized in racks, using load balancing policies.
- However, these policies are limited due to data size and node storage behavior, leading to the need for interconnecting cloud computing.
- Federated Cloud: Cloud environment that utilizes multiple smaller clouds with HDFS storage clusters, comprising one NameNode and multiple DataNodes
- The client can use the federated cloud to communicate with every NameNode to store data across different clusters
- Improved load balancing by storing data through multiple clusters while avoiding overburdening issues.



Diagonally Interleaved Codes

- Burst erasure model that constructs an optimal convolutional code by interleaving data stripes in a diagonal order
    - *c: interval between input messages*
    - *d: total number of symbols in a stripe*
    - *k: number of parity symbols in a stripe*
- An input message is split into a vector of c columns and d-k rows. Blank tables are created between the vector and the message is re-arranged in a diagonal order.
- Next, a systematic block code (e.g. Reed-Solomon) encodes every diagonal group into stripes containing parity symbols
- Diagonally interleaved codes provide extended fault tolerance compared to simpler erasure codes by generating parity data for multiple portions of a data block



	-1	0	1	2	3	4	5	6
1	$B_{1,-1}$	$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{1,5}$	$B_{1,6}$
2	$B_{2,-1}$	$B_{2,0}$	$B_{2,1}$	$B_{2,2}$	$B_{2,3}$	$B_{2,4}$	$B_{2,5}$	$B_{2,6}$
3	$B_{3,-1}$	$B_{3,0}$	$B_{3,1}$	$B_{3,2}$	$B_{3,3}$	$B_{3,4}$	$B_{3,5}$	$B_{3,6}$
P1	Null	Null	Null	$P_1(d_1)$	$P_1(d_2)$	$P_1(d_3)$	$P_1(d_4)$	Null
P2	Null	Null	Null	Null	$P_2(d_1)$	$P_2(d_2)$	$P_2(d_3)$	$P_2(d_4)$

Problem & Challenges

- Problem Definition: How can we achieve high reliability with minimum access cost in Federated clouds?
- Approach: Implement an erasure coding framework which integrates federated cloud storage with metadata techniques

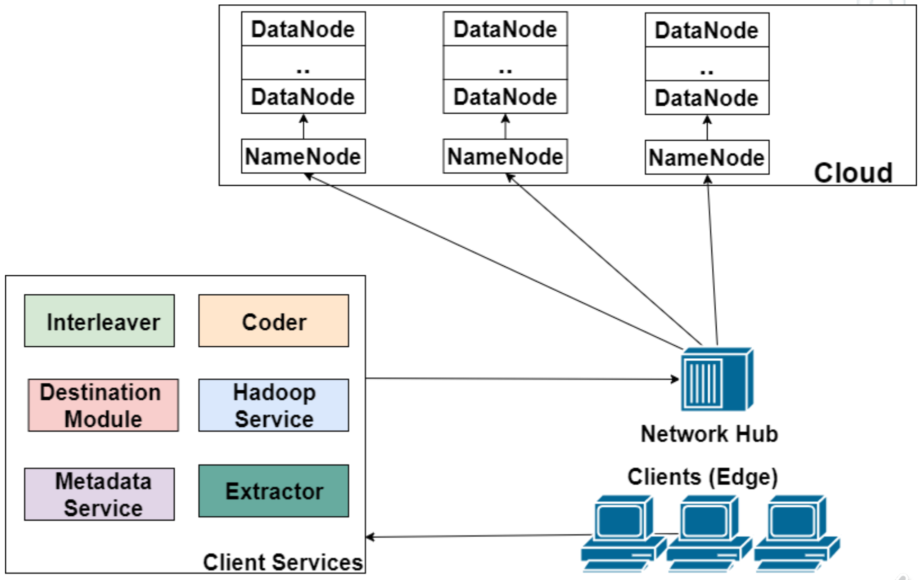
- Challenges:

  1) How can we retrieve data without the need to access a large number of clusters or nodes within the clusters?
  2) How can we enhance the fault tolerance of our system and improve it over simpler erasure codes?
  3) Which load balancing policy should we consider for handling and storing multiple streams of data?

Our Solution: Fed-DIC

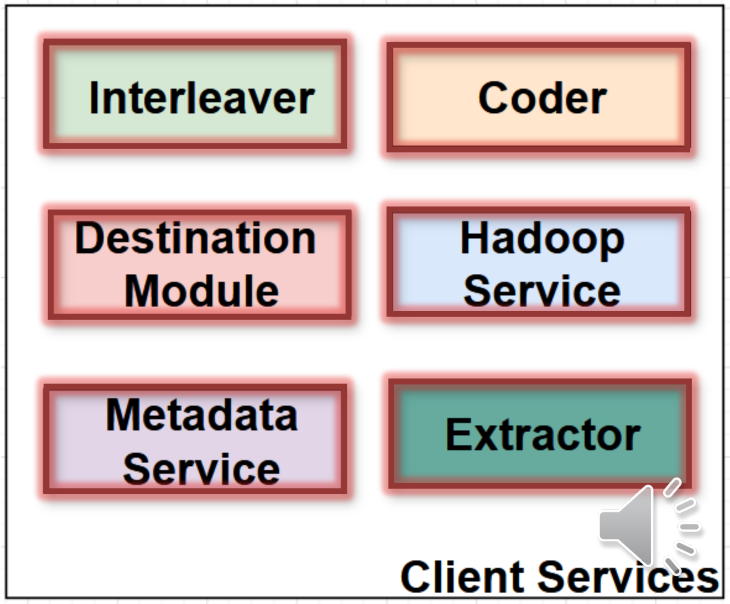
- **Fed-DIC**: **Fed**arated cloud **D**iagonally **I**nterleaved **C**oding
- Utilizes diagonal interleaving and erasure coding on streaming data records in a federated edge cloud environment.
- Supports load balancing by uploading different streams in a rotational order

- Components
  - Edge-side clients
  - Federated cloud
  - Network Hub that connects the clients to the cloud



Client Services

- Interleaver: Arranges input data into a grid and interleaves them into diagonal groups
- Coder: Encodes diagonal groups prior to being uploaded and decodes a diagonal group during the retrieval process
- Destination module: Splits the encoded stripes into batches and configures the destination clusters where the batches will be stored
- Hadoop Service: Communicates with NameNodes of each cluster in order to upload diagonal stripe batches.
- Metadata Service: Creates metadata index for uploaded data directories and provides interface for the user for data retrieval
- Extractor: Searches a received diagonal stripe to extract the requested data record



System Metrics

- Read access cost for a query q: $T_q = l \cdot r_{md} + h \cdot r_h + \sum_{i=1}^{d}((1 - p_i) \cdot t_m)$

- l: number of lines read in metadata file, r_{md}: Reading cost during metadata search
- h: number of accessed clusters, r_h: reading cost on accessing an HDFS cluster
- D: number of chunks in a data stripe, p_i : probability of a chunk being present, t_m : searching delay from a missing chunk

- Overall query storage latency L_q:

$$L_q = T_q + \frac{\sum_{i=1}^{d}(p_i \cdot t_p)}{B} + T_q^{dec}$$

- Total access latency for all Q queries:

$$L_Q = \sum_{q=1}^{Q}(L_q)$$

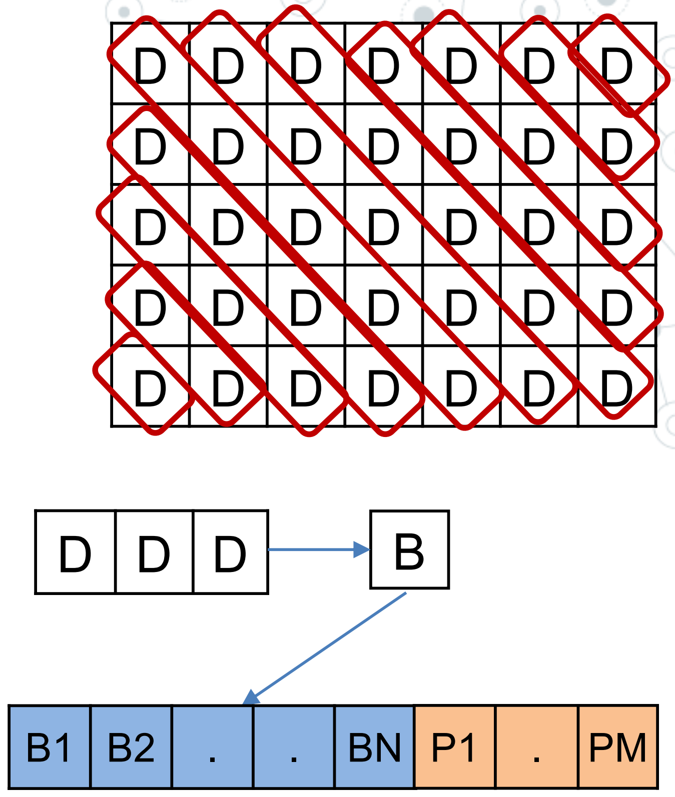
- T_p: chunk transmission time
- B : connection bandwidth
- T^{dec}_q : decoding time for query q
- C : number of encoded diagonal groups
- c_i: a single chunk in a diagonal group

- Data loss percentage:

$$D_C = (1 - \frac{\sum_i^C (p_i \cdot c_i)}{C}) \cdot 100$$

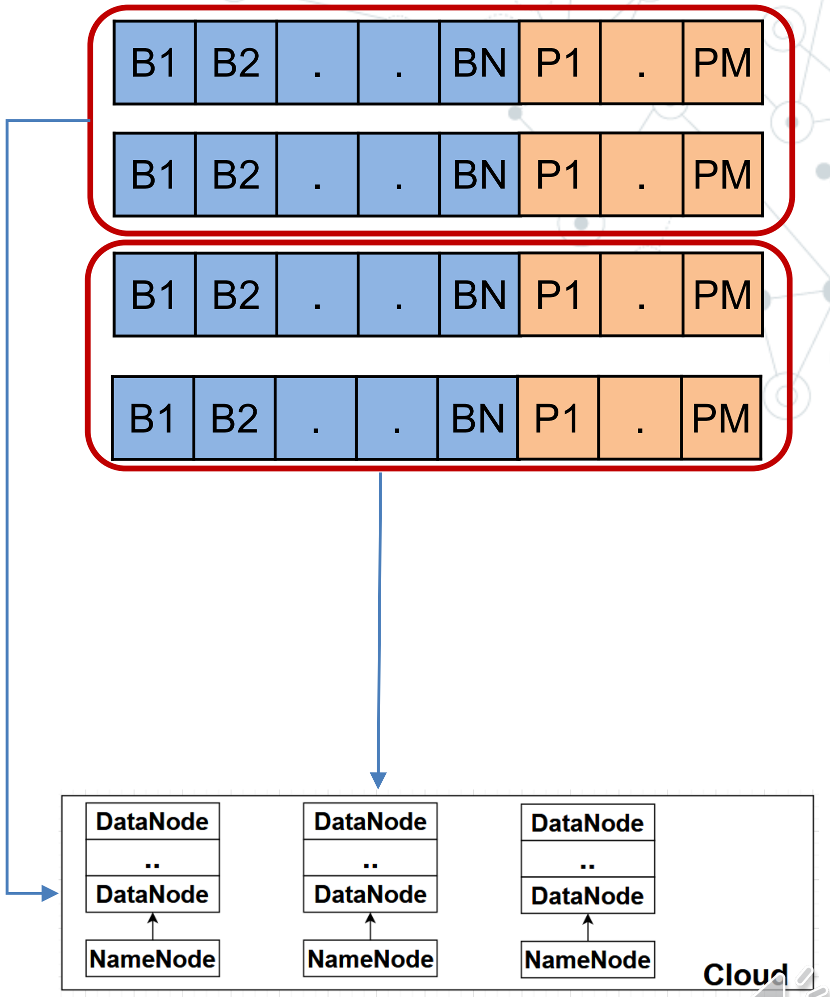
Fed-DIC Operations

- Store data to the federated cloud
  - The input data are trace records that include information for G sensor groups and R days. The data is organized into a grid with R columns and G rows based on the numbers of sensor groups and days.
  - API:
    - Encode(): Groups grid data into diagonal groups, merges these groups into new data blocks and encodes them using RS.
    - Store(): Splits encoded stripes into batch groups, stores them into different clusters within the cloud and creates a metadata file with the locations of the stored data.



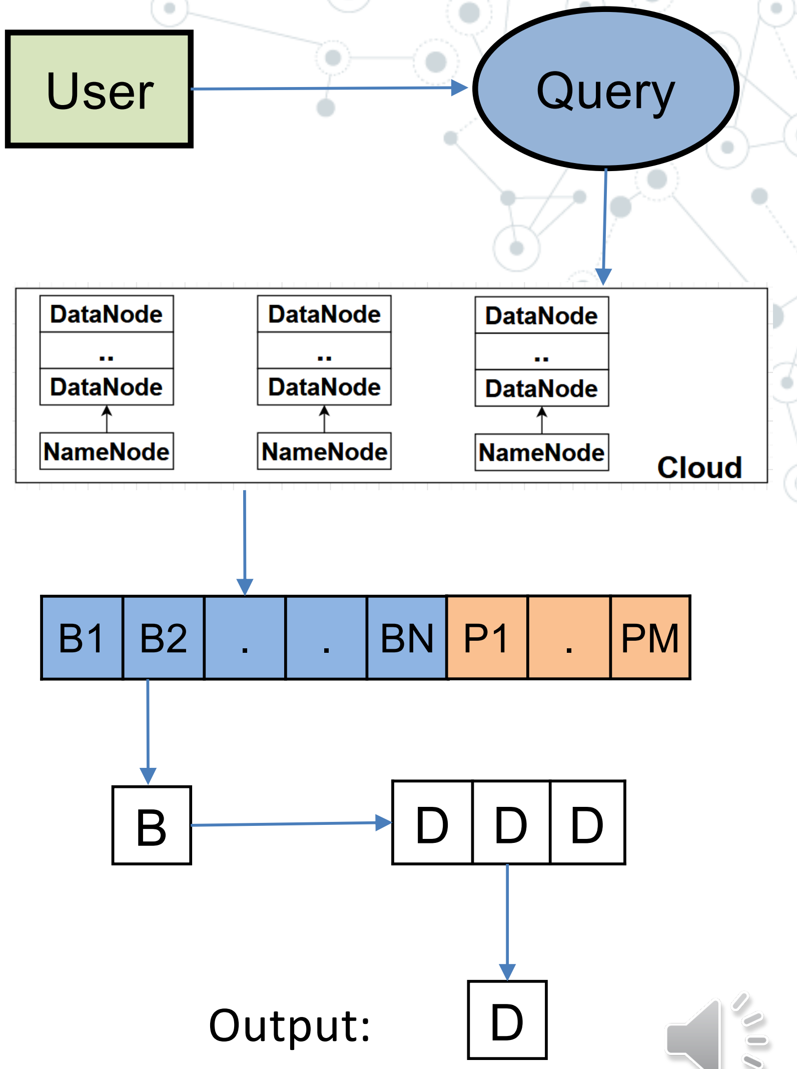
Fed-DIC Operations

- Store data to the federated cloud
  - The input data are trace records that include information for G sensor groups and R days. The data is organized into a grid with R columns and G rows based on the numbers of sensor groups and days.
  - API:
    - Encode(): Groups grid data into diagonal groups, merges these groups into new data blocks and encodes them using RS.
    - Store(): Splits encoded stripes into batch groups, stores them into different clusters within the cloud and creates a metadata file with the locations of the stored data.



Fed-DIC Operations

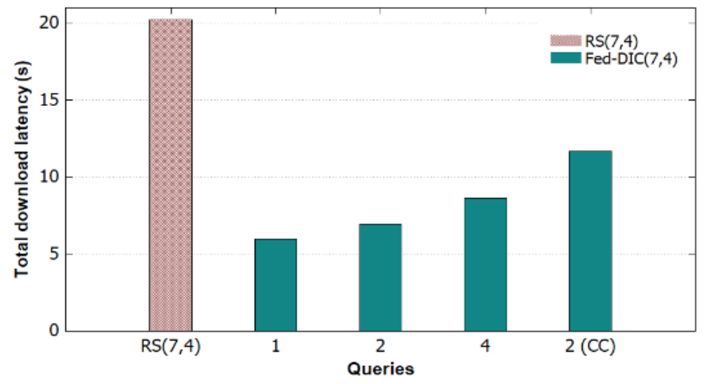
- Retrieve data from the federated cloud
  - The system provides an interface to the user for issuing queries about the day and the sensor group for one or multiple data records. When the queries are created, they are processed by the below API operations:
  - API:
    - Retrieve(): Provides an interface to the user for entering data record queries, searches the metadata file for the diagonal stripe with the requested record and stores temporarily the stripe to the clients.
    - Decode(): Decodes a stripe into its original data and extracts the requested data record from that stripe.

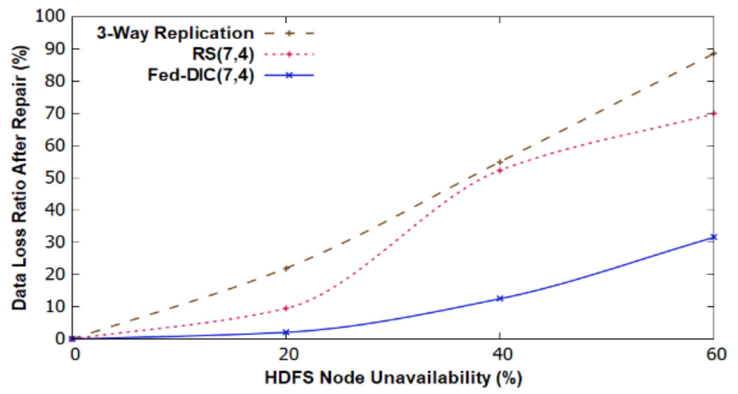


Experiments

- We compared Fed-DIC to 3-way replication and RS(7,4) through a number of experiments
- Client machine specs: Intel i7-7700 4-core 3.5 GHz CPU, 16GB RAM, 1TB disk drive, Windows 10 OS
- Network Hub specs: WAN VPN Router with a data throughput of 100 Mbps and support of 20,000 concurrent connections
- Cloud specs: 4 clusters in Oracle VirtualBox each with 4 VMs, Linux Lubuntu 16.04 OS, Apache Hadoop 3.1.1. We used 2 machines, each running 8 VMs.
- Input data extracted from SCATS sensors that are deployed in Dublin Smart City

Experiments

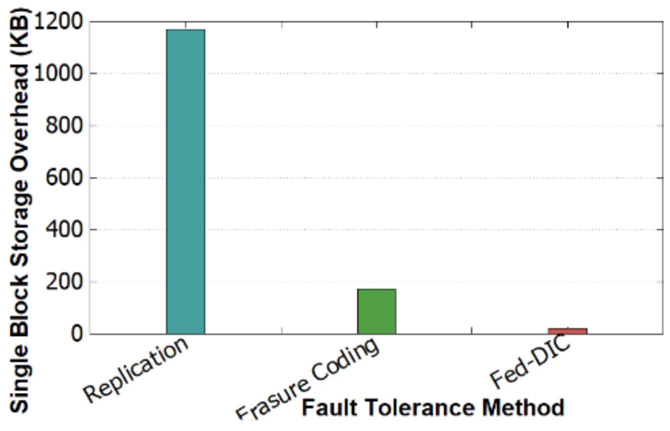




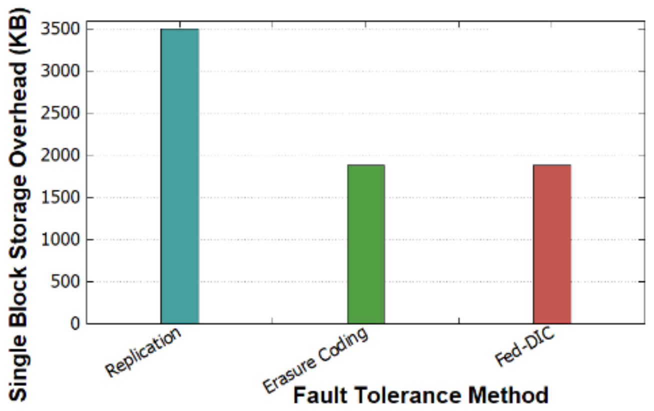
- Total download latency comparison: We attempt to extract a stored data file Reed-Solomon and Fed-DIC using parameters (7,4)
- Unlike Fed-DIC where we can extract a portion of our data, in RS we need to download the entire input data file
- The RS chunks are distributed evenly (3 in first 3 clusters, 2 in last) in order to utilize all of our experimental environment
- With Fed-DIC we can extract up to 4 data records and 2 records across different clusters and achieve up to 60% lower download latency compared to extracting the entire data file with RS

- Data Loss rate between 3 fault tolerance methods
- Even when up to 40% of the nodes are available in the federated cloud, Fed-DIC can maintain a portion of data fully recoverable to the user compared to Replication and RS

Experiments



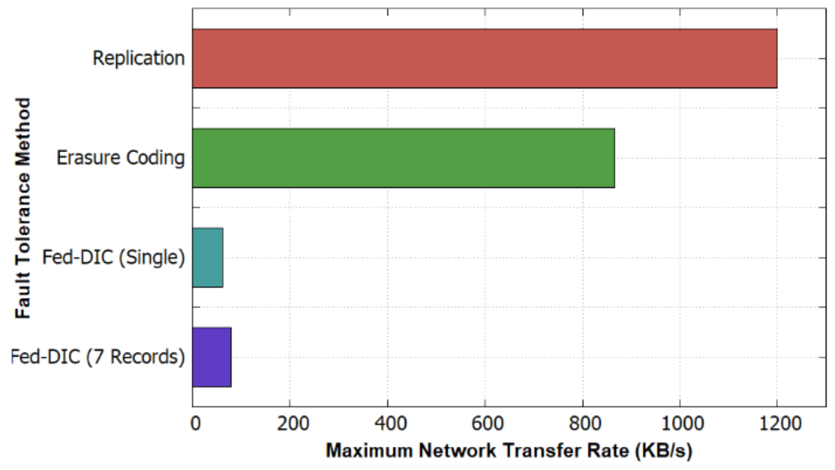
a. Single Chunk



b. All data chunks

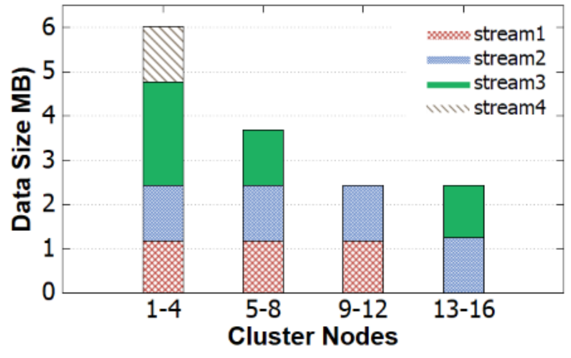
- Storage Overhead between Replication, Erasure Coding and Fed-DIC
- A single chunk generated from erasure coding and Fed-DIC has a significantly smaller storage size compared to a full sized replica created by Replication

Experiments

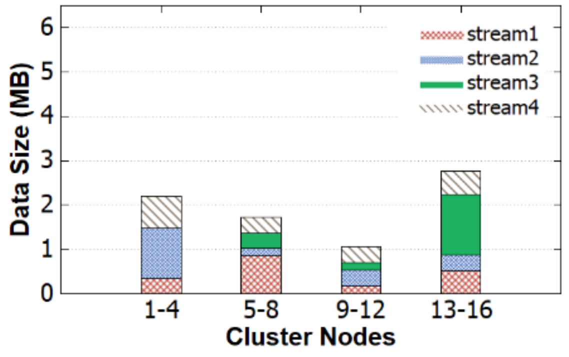


- Maximum Transfer Rate for replication, erasure coding and 2 cases of Fed-DIC (Single record query and 7 record query)
- While Erasure coding and replication overburden the system with high bandwidth rates, Fed-DIC's small data transfers are much less demanding

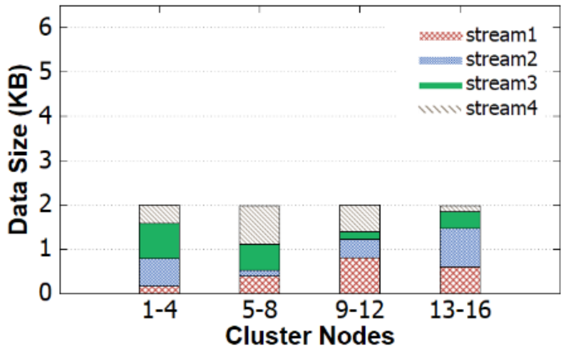
Experiments



a. Replication



b. Reed-Solomon



c. Fed-DIC

- Load balance comparison among the 3 fault tolerance methods
- 4 different streams with similar sizes were uploaded to the cloud with each method
- While Replication and RS place data randomly throughout the clusters, Fed-DIC uploads the streams using the round-robin policy described earlier for balancing the load among the cluster storages

Conclusion

- We presented Fed-DIC, our framework that integrates Diagonal Interleaved Coding with organized storage of the encoded data in a federated cloud environment
- Our experimental evaluations illustrate the benefits of our framework compared to state-of-the-art fault tolerance methods in terms of total read access latency, data loss percentage, maximum network transfer rate, storage overhead and load balancing
- For future work, we plan to deploy Fed-DIC in a federated environment with different types hardware and equipment

Fed-DIC: Diagonally Interleaved Coding in a Federated Cloud Environment

Giannis Tzouros
Department of Informatics
Athens University of Economics and Business

Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business