

IPFS & Sia

Presentazione progetto esame reti P2P

Alessandro Serra mat.891061

Andrea Longo mat.902517

IPFS

IPFS è uno storage distribuito che si pone come obiettivo quello di fornire un file system globale e decentralizzato per l'archiviazione, condivisione e il recupero di documenti e file di qualsiasi tipo e origine. Mira a sostituire HTTP.

Vantaggi:

- Nessuna dipendenza dai server
- Riduzione di costo
- Riduzione della censura dell'ISP
- Storicità dei dati memorizzati sulla rete



IPFS - Identità e rete

I nodi sono identificati dal NodeId (hash crittografato della chiave pubblica) creato con il cripto puzzle statico di S/Kademlia. I nodi memorizzano le loro chiavi pubbliche e private.

Si utilizza il multihash in modo da poter individuare la migliore funzione hash da utilizzare.

La connessione avviene se l'hash della chiave pubblica è uguale al NodeId.

La comunicazione avviene tra centinaia di nodi. Viene fornita:

- Affidabilità (attraverso uTP o SCTP), connettività (usando anche tecniche NAT ICE), integrità (usando facoltativamente un checksum hash) e autenticità (usando facoltativamente HMAC con la chiave pubblica del mittente).

IPFS memorizza gli indirizzi come stringhe di byte in formato multiaddr: esprime gli indirizzi e i loro protocolli, incluso il supporto per l'incapsulamento.

IPFS - Routing e scambio di blocchi

I nodi IPFS richiedono un sistema di routing in grado di trovare indirizzi di rete di altri peer e peer in grado di servire oggetti particolari.

Vengono usate le DHT. IPFS DHT distingue:

- Piccoli valori (uguali o inferiori a 1 KB) vengono memorizzati direttamente sul DHT.
- Per valori maggiori, il DHT memorizza i riferimenti, che sono i Nodeld dei peer che possono servire il blocco.

La distribuzione dei dati avviene scambiando dei blocchi con i peer usando un protocollo ispirato a BitTorrent: BitSwap.

Due liste:

- Want_list: ciò che voglio acquisire.
- Have_list: ciò che ho da offrire.

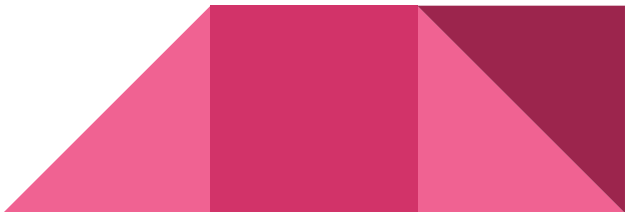


IPFS - Scambio di blocchi/2

Due situazioni possibili:

- Scambio complementare: i nodi hanno ciò che vogliono (e lo scambiano).
- Incentivo a memorizzare: un nodo non ha ciò che i suoi coetanei vogliono e quindi cerca i pezzi con una priorità inferiore rispetto a ciò che il nodo stesso desidera. Ciò incentiva i nodi a memorizzare e diffondere pezzi.


Credito e strategie:

1. I peer tengono traccia del loro equilibrio con altri nodi.
 2. I peer inviano blocchi ai peer debitori, secondo una funzione probabilistica che diminuisce all'aumentare del debito.
 3. Massimizzare le prestazioni per il nodo e l'intero scambio.
 4. Impedire ai freeloader di sfruttare e degradare lo scambio.
 5. Essere efficace e resistente ad altre strategie sconosciute.
- 

IPFS - Ledger e protocollo

I nodi bitswap mantengono un registro(ledger) dove sono registrati tutti gli scambi. Ad inizio connessione vengono scambiati i registri: se le informazioni non corrispondono allora il registro viene azzerato assieme al debito e al credito.

Il protocollo seguito da BitSwap è:

1. Apertura: i peer inviano i registri fino a quando non sono d'accordo.
 2. Invio: i peer si scambiano le liste (want_list, have_list).
 3. Chiusura: i peer disattivano la connessione.
 4. Ignorato(speciale): un peer viene ignorato (per la durata di un timeout) se la strategia di un nodo evita l'invio.
- 

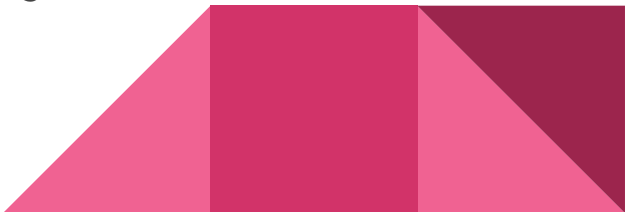
IPFS - Merkle DAG

IPFS collega tutto ciò che è stato descritto attraverso il Merkle DAG, un grafo aciclico diretto in cui i collegamenti tra gli oggetti sono gli hash crittografici delle destinazioni. Tale struttura permette di sfruttare le caratteristiche di:

- Path:

```
# format
/ipfs/<hash-of-object>/<name-path-to-object>

# example
/ipfs/XLYkgq61DYaQ8NhkcqyU7rLcnSa7dSHQ16x/foo.txt xt
```
- Pinning: contenuti bloccati all'interno della rete, conservati nella memoria locale del nodo.
- Crittografia: è possibile avere contenuti e collegamenti crittografati. Senza la chiave di crittografia non si può risalire al contenuto/collegamento.



IPFS - Gestione dei file

IPFS definisce anche dei modelli di oggetti simile a Git:

- Blocco: un blocco di dati di dimensioni variabili.
- Elenco: una raccolta di blocchi o altri elenchi.
- Albero: una raccolta di blocchi, elenchi o altri alberi.
- Commit: un'istantanea nella cronologia delle versioni di un albero.

```
{
  "data": ["blob", "list", "blob"],
  // lists have an array of object types as data
  "links": [
    { "hash": "XLYkgq61DYaQ8NhkcqyU7rLcnSa7dSHQ16x",
      "size": 189458 },
    { "hash": "XLHBNmRQ5sJJrdMPuu48pzeyTtRo39tNDR5",
      "size": 19441 },
    { "hash": "XLWVQDqxo9Km9zLyquoC9gAP8CL1gWnHZ7z",
      "size": 5286 }
  ]
  // lists have no names in links
}
```

```
> ipfs file-cat <ccc111-hash> --json
{
  "data": {
    "type": "tree",
    "date": "2014-09-20 12:44:06Z",
    "message": "This is a commit message."
  },
  "links": [
    { "hash": "<ccc000-hash>",
      "name": "parent", "size": 25309 },
    { "hash": "<ttt111-hash>",
      "name": "object", "size": 5198 },
    { "hash": "<aaa111-hash>",
      "name": "author", "size": 109 }
  ]
}
```


JS IPFS



Il progetto è stato sviluppato utilizzando JS-IPFS, librerie IPFS per NodeJS.

```
const node = await IPFS.create()

const data = 'Hello, <YOUR NAME HERE>'

// add your data to IPFS - this can be a string, a Buffer,
// a stream of Buffers, etc
const files = await node.add(data)

// 'hash', known as CID, is a string uniquely addressing the data
// and can be used to get it again. 'files' is an array because
// 'add' supports multiple additions, but we only added one entry
console.log(files[0].hash)
```

Le librerie sono state importate usando lo strumento npm.

La libreria permette di accedere a tutte le funzioni conosciute di IPFS (libp2p, daemon, DAG Api) attraverso l'utilizzo di NodeJS.

```
const node = await IPFS.create()

const data = await node.cat('QmPChd2hVbrJ6bfo3WBcTW4iZnpHm8TEzWkLHmLpXhF68A')

// data is returned as a Buffer, convert it back to a string
console.log(data.toString())
```

Sia

Sia è una piattaforma di cloud storage decentralizzata, che sfrutta la capacità di disco rigido non utilizzata e messa a disposizione dai propri utenti.

Questo sistema è alimentato da una propria blockchain e criptovaluta chiamata appunto Siacoin.

Il progetto è open-source ed è stato creato nel 2014 dal team di NebulousLabs.



Sia - Sicurezza e disponibilità

I file caricati per poter essere caricati all'interno della rete Sia vengono:

Divisi in più segmenti, solitamente ciascun file viene suddiviso in 30 segmenti ma questo può variare a seconda delle sue dimensioni, questa operazione viene svolta utilizzando un Codice Reed-Solomon.

Ciascun segmento viene criptato secondo l'algoritmo XChaCha20.

I vari segmenti criptati vengono distribuiti all'interno della rete in maniera ridondante, in modo tale da rendere i file quanto più reperibili possibili anche in mancanza di disponibilità di uno dei nodi.

Sia - Ruoli nella rete

Nella rete Sia ogni nodo presente può svolgere un ruolo preciso rispetto agli altri, fra cui si distinguono i seguenti:

- Renter: nodo che vuole caricare i propri file all'interno della rete
- Host: nodo che fornisce il proprio spazio disco per immagazzinare i dati

L'Host propone al Renter un prezzo da pagare per poter usufruire del proprio spazio messo a disposizione, e l'accordo viene registrato all'interno della blockchain di Sia tramite smart-contracts.



Sia - Proof of Storage

Periodicamente gli Host devono fornire una Proof of Storage dei file che essi mantengono, questo per poter ricevere l'effettivo pagamento (fee) accordato con i Renter tramite smart-contract.

Quanti più segmenti di un file un Host possiede, più probabilità avrà di ricevere le intere fee dello smart-contract riguardo quei file.

Questo scenario non è comunque plausibile perchè la rete Sia fa in modo che un Host non sia mai in possesso di tutti i segmenti di un file.



Skynet - Testnet Sia

Skynet nasce come portale di testnet per sviluppatori, in cui è possibile effettuare Upload di qualsiasi file sulla rete Sia, e di recuperare anche i file tramite link univoci chiamati Skylink.

La necessità della creazione di questa rete nasce dal fatto che il creatori vogliono rendere disponibile questo cloud storage anche a tutti gli utenti che non vogliono essere direttamente partecipi nella rete come nodi Renter.



Skynet - Linguaggi supportati

La testnet di Skynet è accessibile ed utilizzabile tramite vari linguaggi di programmazione/scripting, tra cui:

- Node.js
- Python
- Go

Attraverso le loro librerie è possibile al momento effettuare le operazioni essenziali di Upload file e Download di file tramite Skylink.




Skynet - Skylink

Attraverso degli algoritmi di codifica, partendo dalle informazioni del file all'interno della rete Sia, e tradotti in informazioni su 16 bit, vengono creati i cosiddetti Skylink, identificatori univoci per i file caricati all'interno della rete Sia ed utilizzabili per la condivisione online, e si presentano nella forma:

[sia://_AapiCnm6kx-5UQtIGIHwOcuJmS-Xk44JpcoG4kPOSly2Q](#)

Ogni volta che un file già presente nella rete sia viene caricato, lo Skylink generato rimarrà uguale a quello precedentemente ottenuto dallo stesso file.



Skynet - Javascript

Per utilizzare la testnet di Sia offerta da Skynet abbiamo fatto affidamento alle librerie Node.js offerte dal team di Nebulous e disponibili in open-source nella loro repository:

<https://github.com/NebulousLabs/nodejs-skynet>

```
1  const skynet = require('@nebulous/skynet');
2
3  (async () => {
4      // upload
5      const skylink = await skynet.UploadFile(
6          './src.jpg',
7          skynet.DefaultUploadOptions
8      );
9      console.log(`Upload successful, skylink: ${skylink}`);
10
11     // download
12     await skynet.DownloadFile(
13         './dst.jpg',
14         skylink,
15         skynet.DefaultDownloadOptions
16     );
17     console.log('Download successful');
18 })()
```

Test effettuati

Si è scelto di effettuare 3 test per ciascuna delle due reti (IPFS e Sia). Nello specifico:

- Test di caricamento di dati provenienti dalle linee bus della città di Rio de Janeiro: si partiva dalla fase di lettura di alcuni file csv inerenti la simulazione dei percorsi degli autobus seguita dal successivo caricamento sulla rete dei dati estratti.
- Test di caricamento di big file: caricamento sulle due reti di alcuni file di dimensioni maggiori.
- Download dei dati inseriti nella fase precedente: nello specifico si è effettuato il download attraverso un secondo dispositivo che non ha una copia locale dei file.



Grafici caricamento dati bus

Di seguito verranno proposti i grafici relativi alle latenze ottenute durante il caricamento dei dati inerenti i bus.

Sono proposti due grafici per ogni bus:

- Delle latenze: rappresenta tutte le latenze ottenute per ogni test effettuato.
 - Asse X: id relativo al counter del bus.
 - Asse Y: tempo di esecuzione, latenza in ms.
- Delle aggregazioni: sono rappresentate le informazioni circa latenza media, latenza accumulata e deviazione standard per ogni bus.
 - Asse X: id relativo al test effettuato.
 - Asse Y: tempo di esecuzione, latenza in ms.

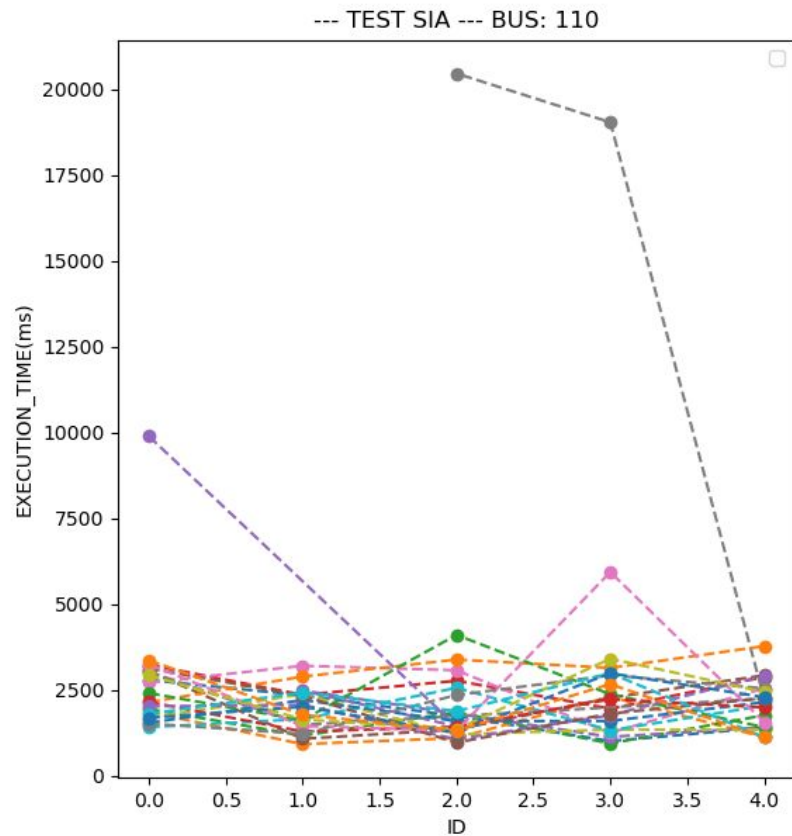
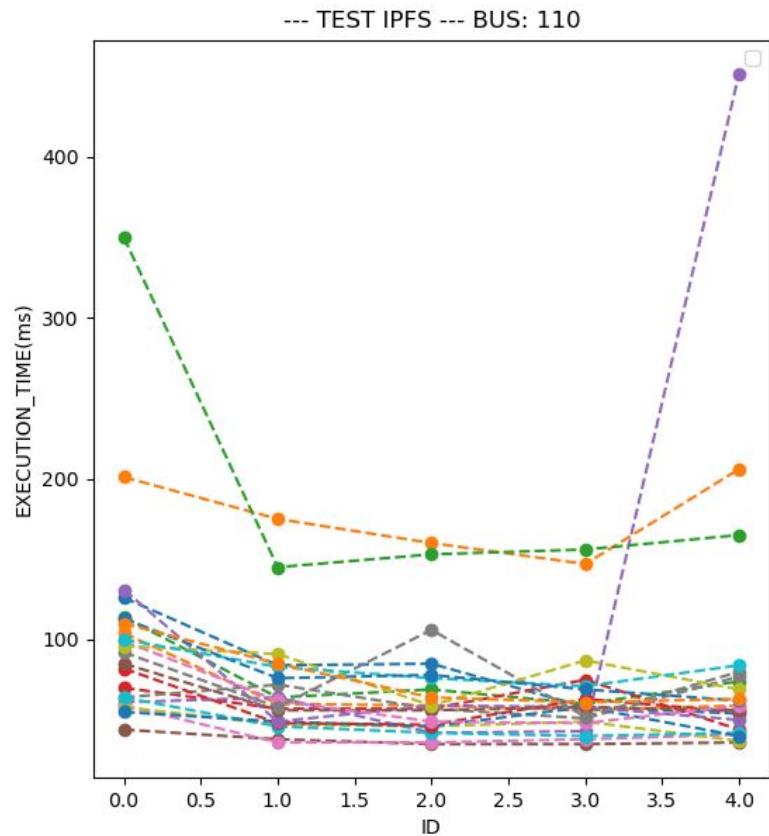


Grafici caricamento dati bus - Legenda

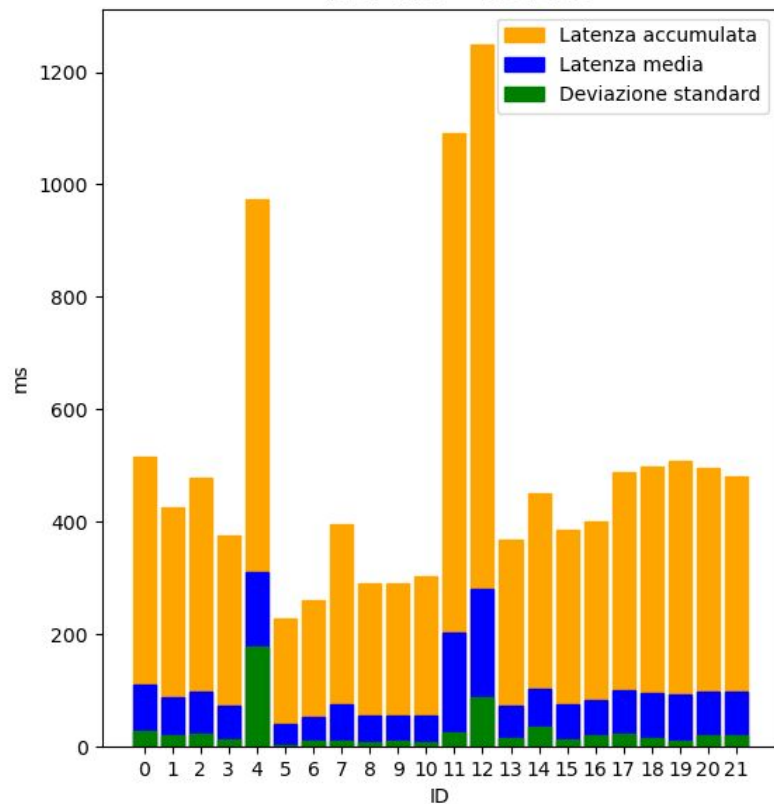
Legenda per il
grafico delle
aggregazioni.

0	Graph for: .\tests\2020-02-28T11_09_41.371Z
1	Graph for: .\tests\2020-02-28T12_57_20.350Z
2	Graph for: .\tests\2020-02-28T13_30_46.749Z
3	Graph for: .\tests\2020-03-07T15_33_20.335Z
4	Graph for: .\tests\2020-03-07T22_47_00.440Z
5	Graph for: .\tests\2020-03-08T10_08_59.666Z
6	Graph for: .\tests\2020-03-08T18_39_27.848Z
7	Graph for: .\tests\2020-03-09T09_52_42.723Z
8	Graph for: .\tests\2020-03-11T18_47_28.299Z
9	Graph for: .\tests\2020-03-11T22_44_41.412Z
10	Graph for: .\tests\2020-03-12T23_47_10.647Z
11	Graph for: .\tests\2020-03-13T16_14_36.930Z
12	Graph for: .\tests\2020-03-15T21_27_33.292Z
13	Graph for: .\tests\2020-03-16T10_31_14.751Z
14	Graph for: .\tests\2020-03-16T14_37_55.391Z
15	Graph for: .\tests\2020-03-17T10_37_53.035Z
16	Graph for: .\tests\2020-03-17T15_51_26.316Z
17	Graph for: .\tests\2020-03-17T22_14_28.230Z
18	Graph for: .\tests\2020-03-18T10_08_18.500Z
19	Graph for: .\tests\2020-03-18T14_45_47.839Z
20	Graph for: .\tests\2020-03-18T22_32_26.772Z
21	Graph for: .\tests\2020-03-19T08_04_54.946Z

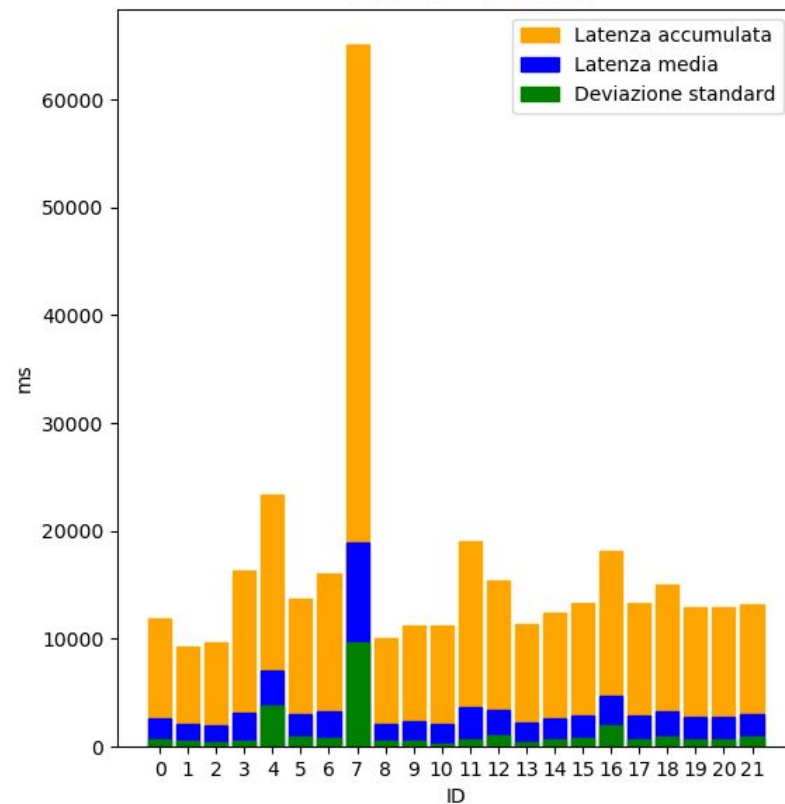
Legenda per il
grafico delle
latenze.



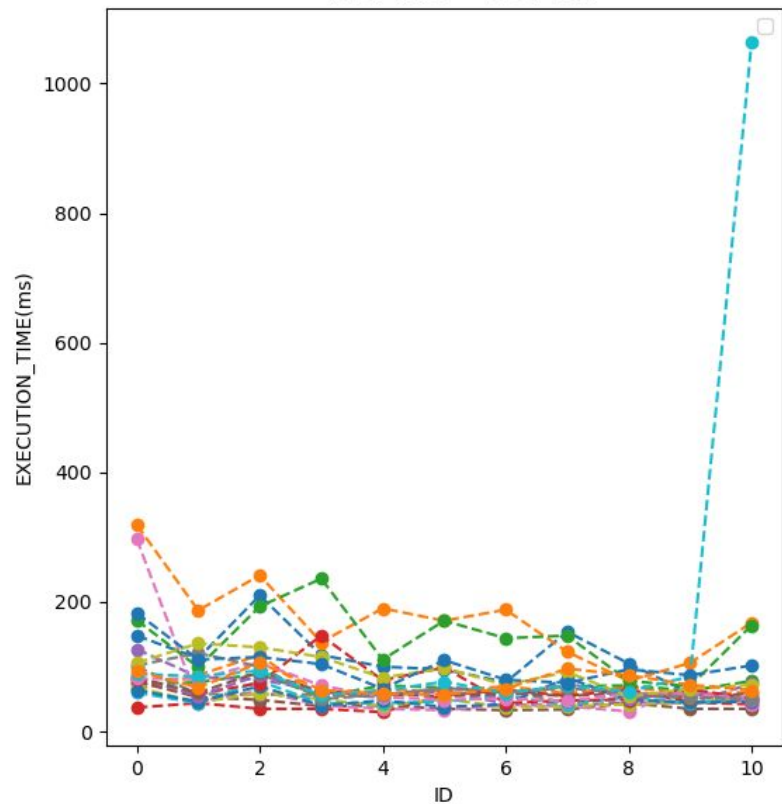
--- TEST IPFS --- BUS: 110



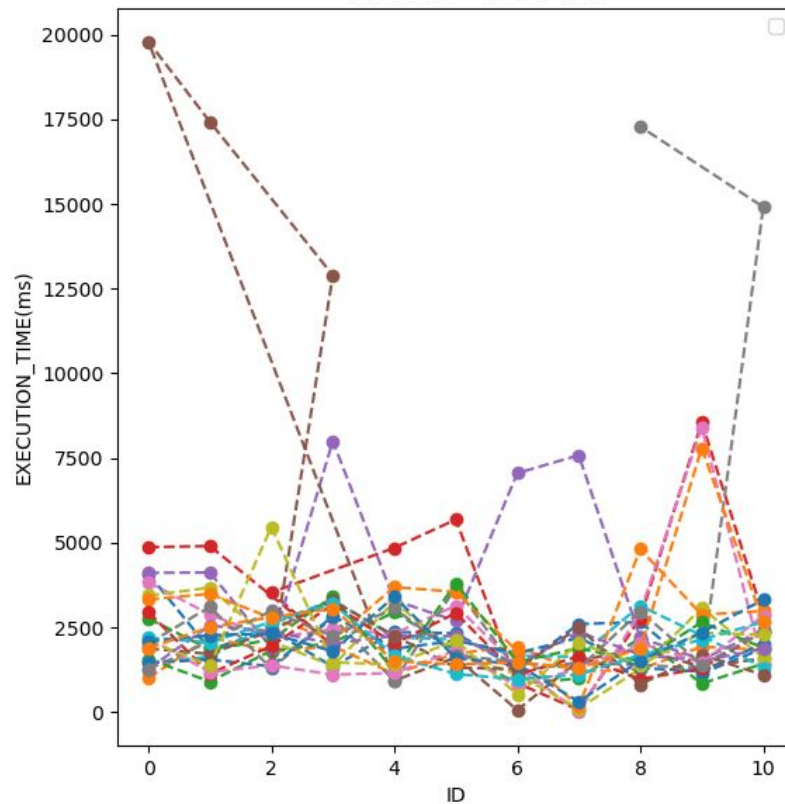
--- TEST SIA --- BUS: 110



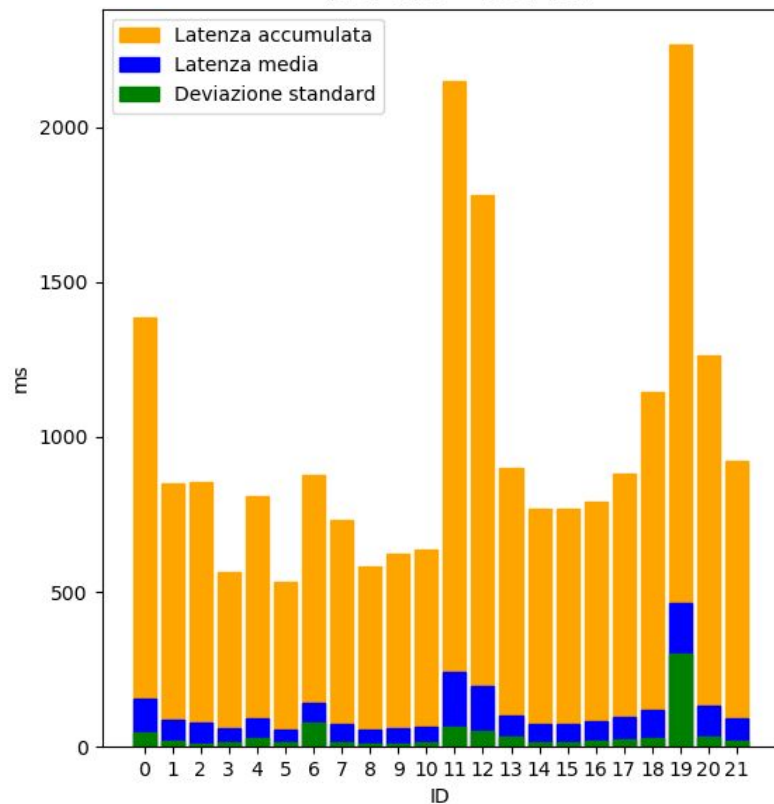
--- TEST IPFS --- BUS: 422



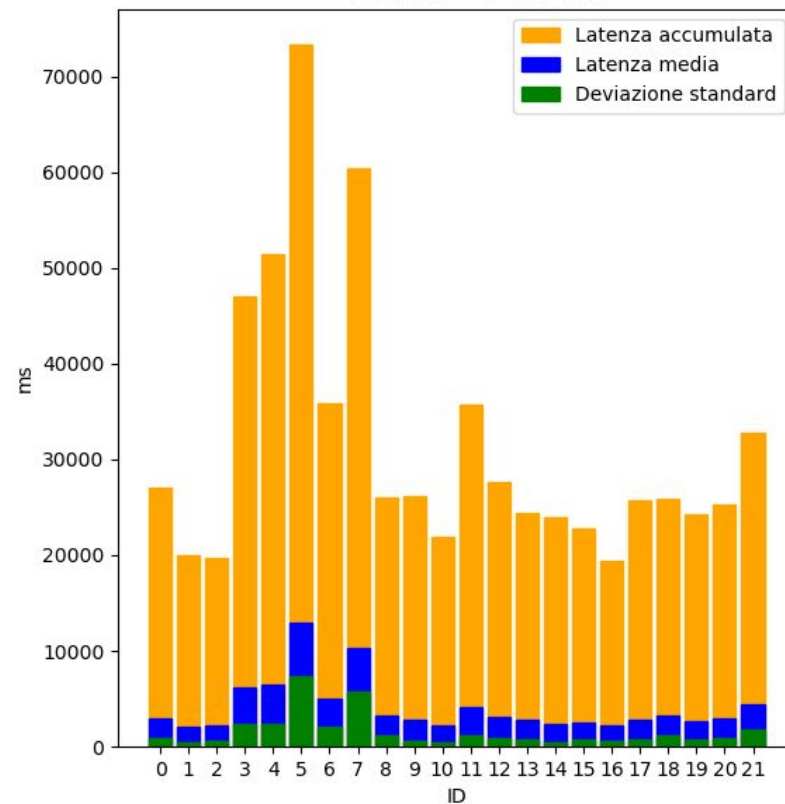
--- TEST SIA --- BUS: 422

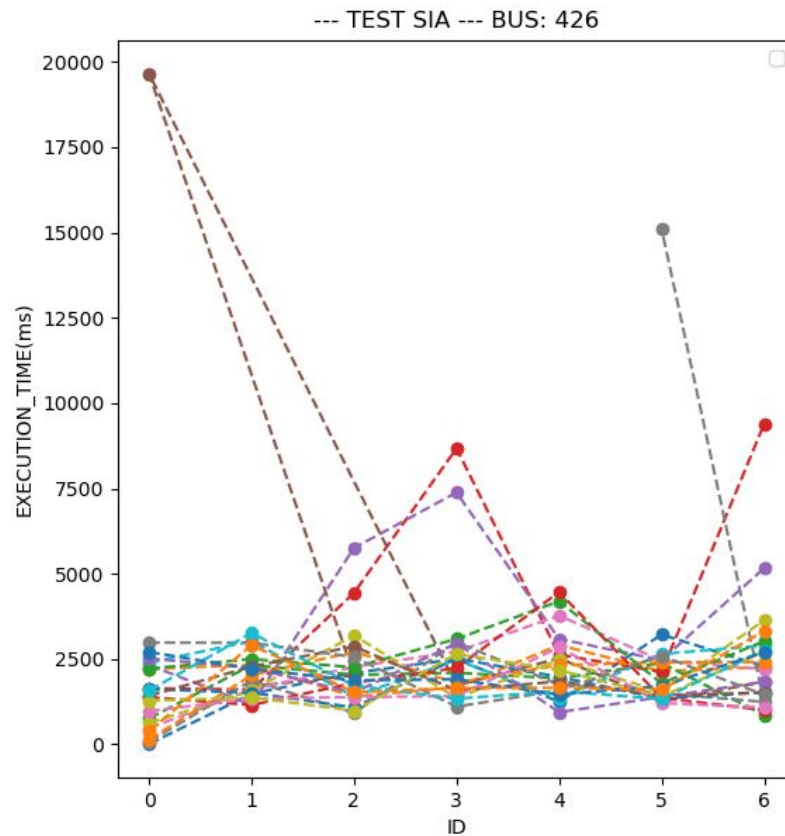
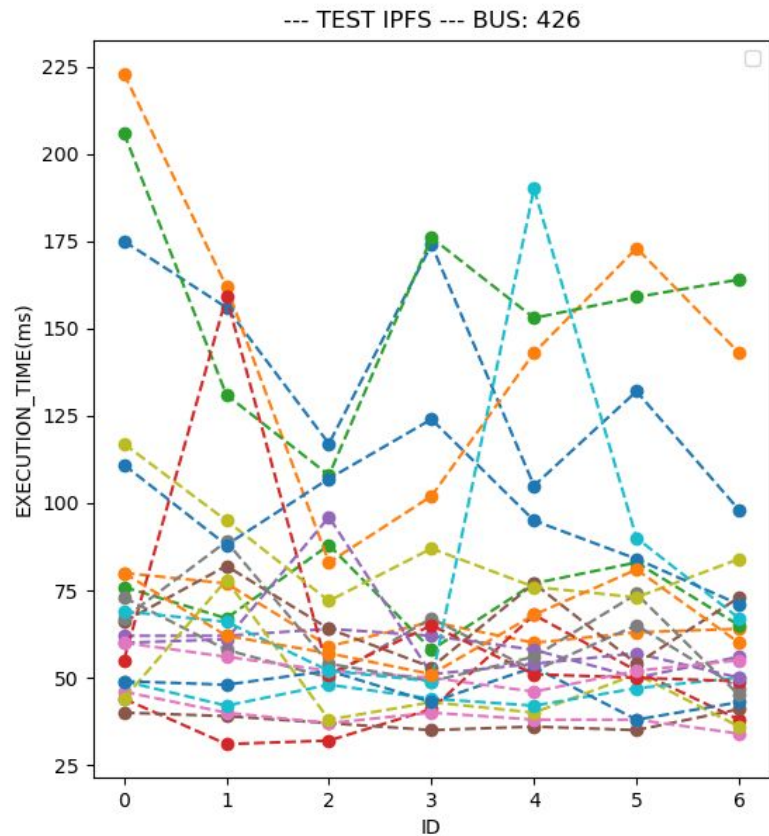


--- TEST IPFS --- BUS: 422

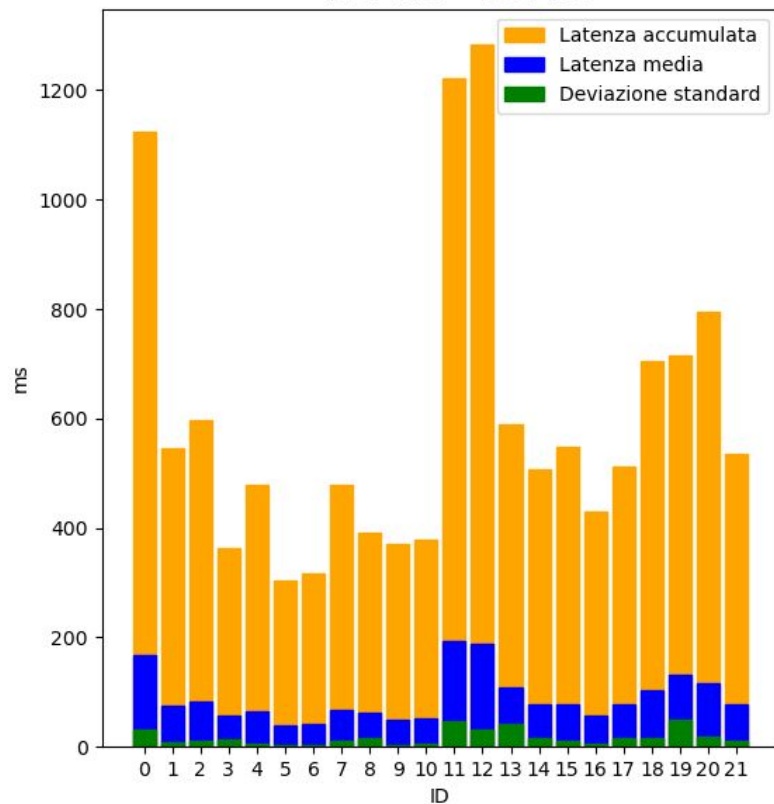


--- TEST SIA --- BUS: 422

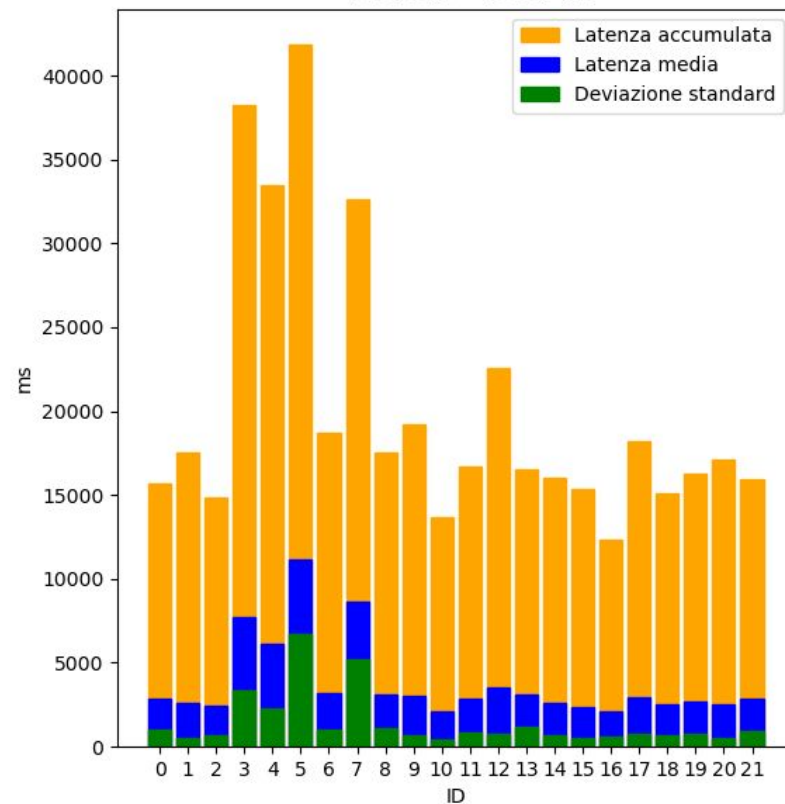


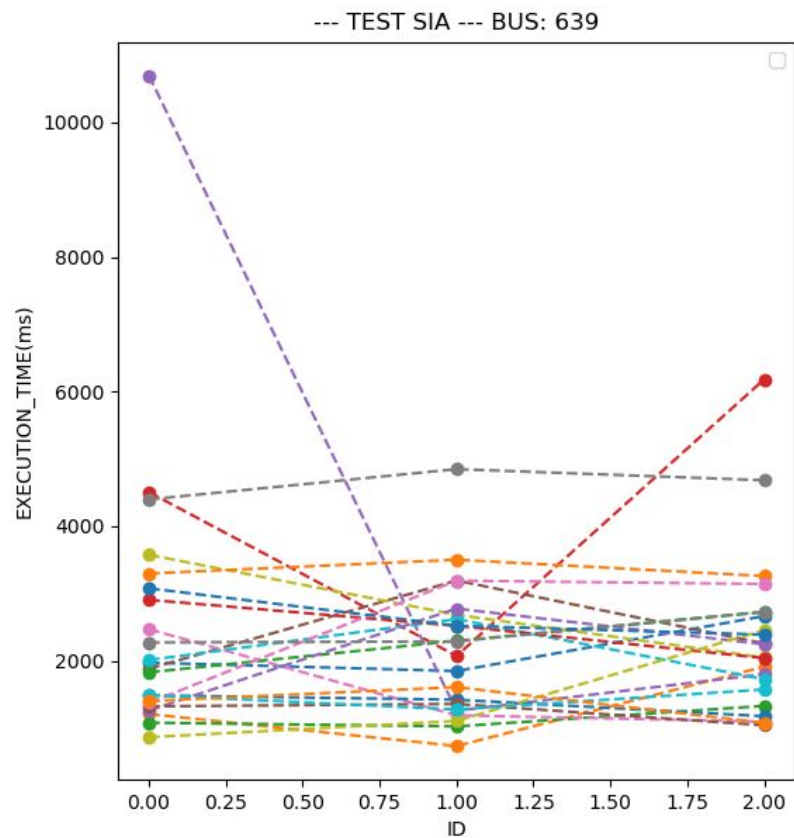
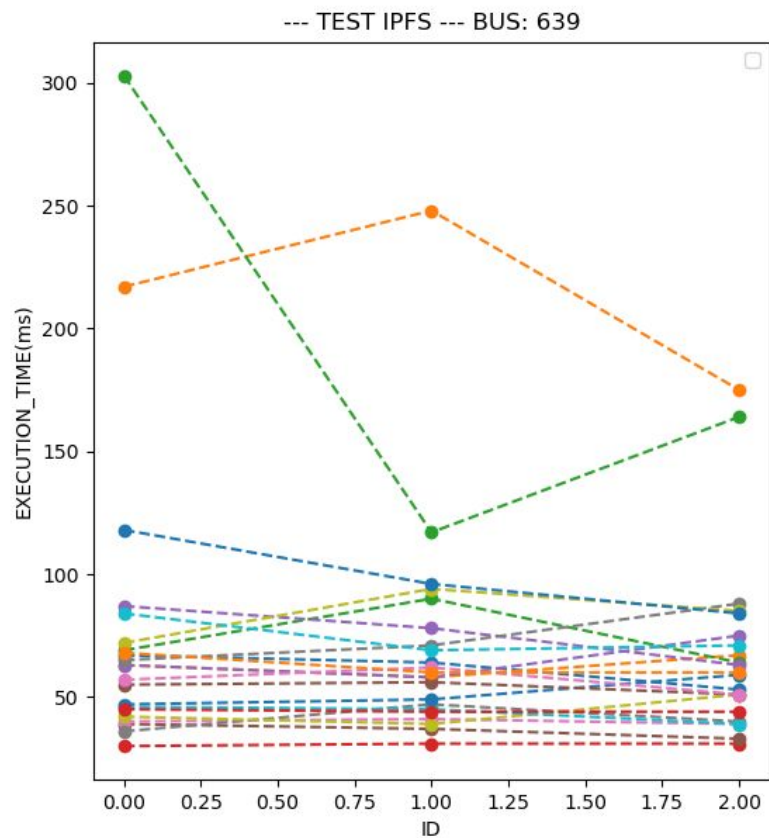


--- TEST IPFS --- BUS: 426

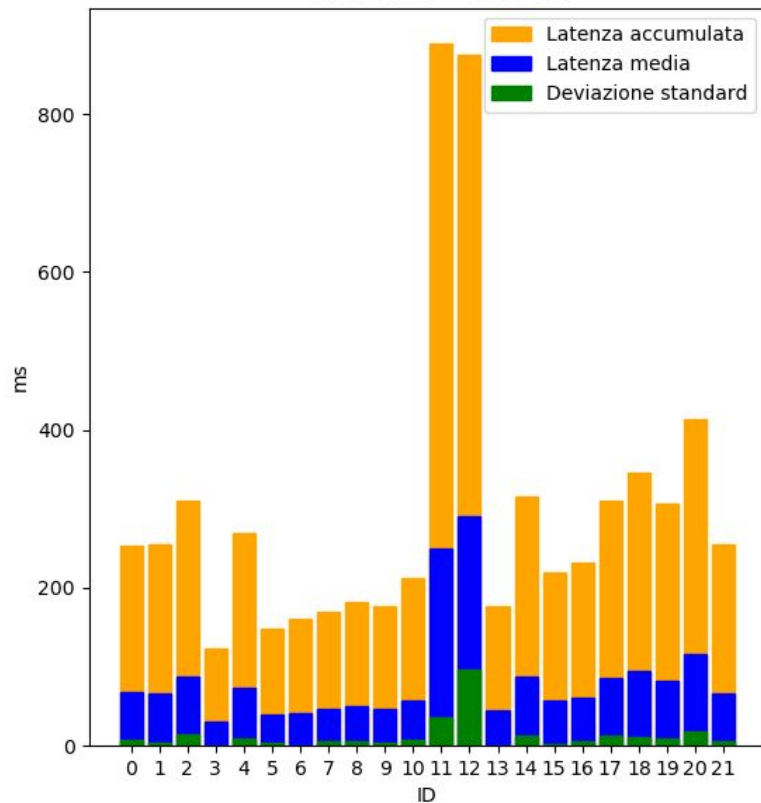


--- TEST SIA --- BUS: 426





--- TEST IPFS --- BUS: 639



--- TEST SIA --- BUS: 639

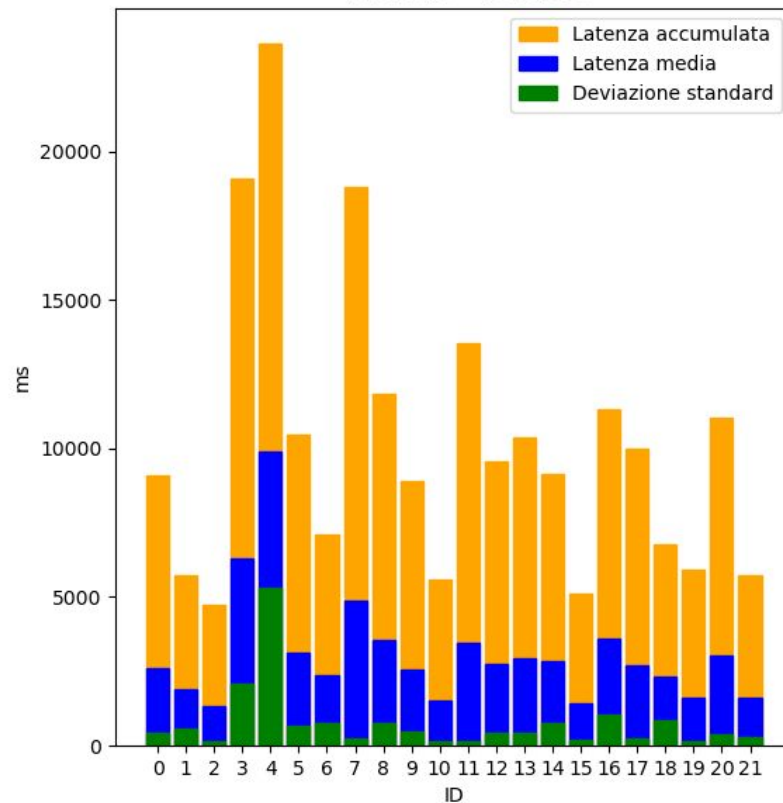


Grafico caricamento big file

Di seguito verrà proposto il grafico relativo alle latenze ottenute durante il caricamento di 4 file selezionati dalle dimensioni maggiori. Nello specifico:

- 1.txt: documento di testo (dimensioni: 6 byte)
- 2.jpg: immagine (dimensioni: 1,49 MB)
- 3.pdf: documento pdf (dimensioni 222 kB)
- 4.mp4: video (dimensioni 9,24 MB).

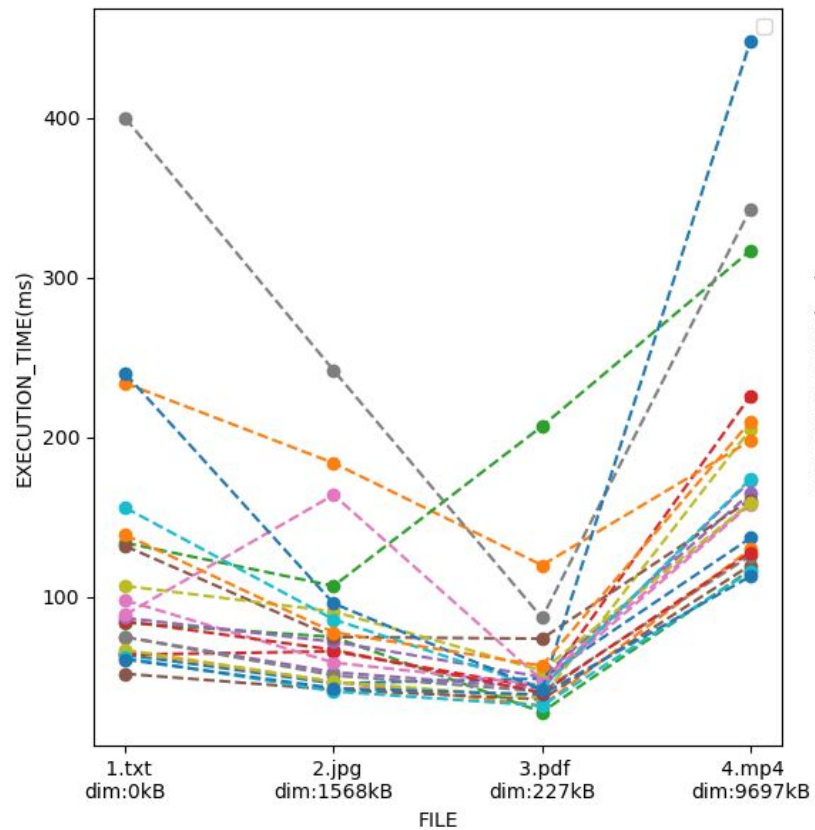
Sono proposti due grafici per ogni file:

- Delle latenze: rappresenta tutte le latenze ottenute per ogni test effettuato.
 - Asse X: nome e dimensione del file.
 - Asse Y: tempo di esecuzione, latenza in ms.
- Delle aggregazioni: sono rappresentate le informazioni circa latenza media, latenza accumulata e deviazione standard per ogni file.
 - Asse X: id relativo al file (ID=1 -- 1.txt, ID=2 -- 2.jpg, ID=3 -- 3.pdf, ID=4 -- 4.mp4).
 - Asse Y: tempo di esecuzione, latenza in ms.

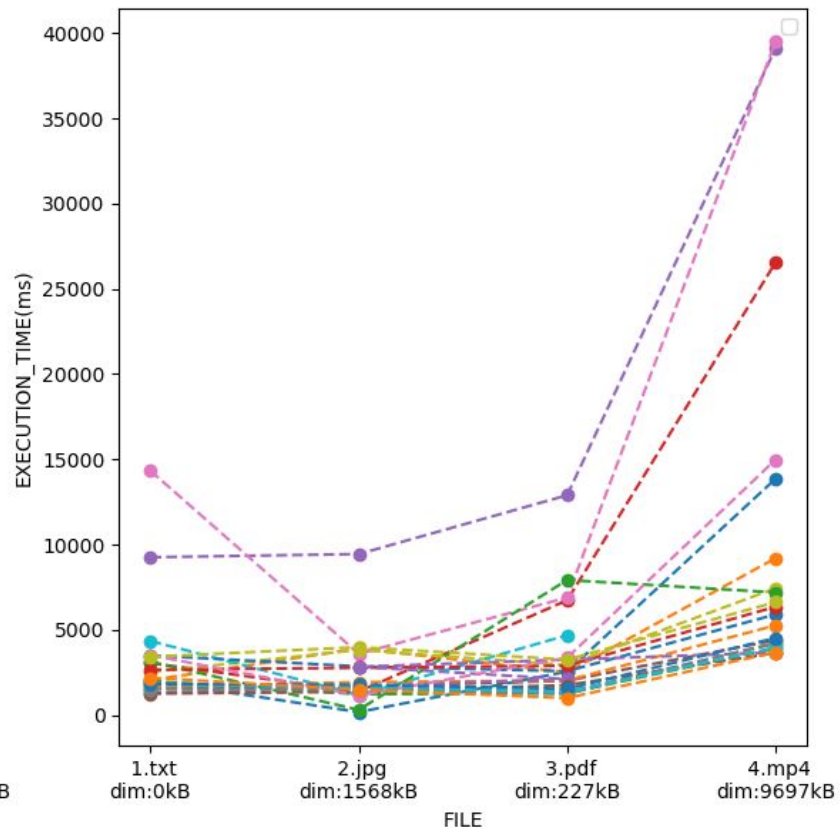
Grafici caricamento big file - Legenda

Graph for: .\bigfiletests\tests\2020-02-28T16_31_37.401Z
Graph for: .\bigfiletests\tests\2020-02-28T16_32_25.774Z
Graph for: .\bigfiletests\tests\2020-02-28T16_33_23.111Z
Graph for: .\bigfiletests\tests\2020-03-07T15_49_18.278Z
Graph for: .\bigfiletests\tests\2020-03-07T23_04_23.788Z
Graph for: .\bigfiletests\tests\2020-03-08T10_32_03.269Z
Graph for: .\bigfiletests\tests\2020-03-08T19_17_56.951Z
Graph for: .\bigfiletests\tests\2020-03-09T10_18_14.946Z
Graph for: .\bigfiletests\tests\2020-03-09T18_18_45.841Z
Graph for: .\bigfiletests\tests\2020-03-09T23_02_49.084Z
Graph for: .\bigfiletests\tests\2020-03-10T09_37_51.129Z
Graph for: .\bigfiletests\tests\2020-03-10T15_42_38.698Z
Graph for: .\bigfiletests\tests\2020-03-15T21_45_39.926Z
Graph for: .\bigfiletests\tests\2020-03-16T11_08_44.216Z
Graph for: .\bigfiletests\tests\2020-03-16T15_10_02.380Z
Graph for: .\bigfiletests\tests\2020-03-17T11_01_40.604Z
Graph for: .\bigfiletests\tests\2020-03-17T16_21_45.652Z
Graph for: .\bigfiletests\tests\2020-03-17T23_00_14.889Z
Graph for: .\bigfiletests\tests\2020-03-18T10_38_44.517Z
Graph for: .\bigfiletests\tests\2020-03-18T15_01_40.881Z
Graph for: .\bigfiletests\tests\2020-03-18T23_00_10.432Z
Graph for: .\bigfiletests\tests\2020-03-19T08_51_27.035Z

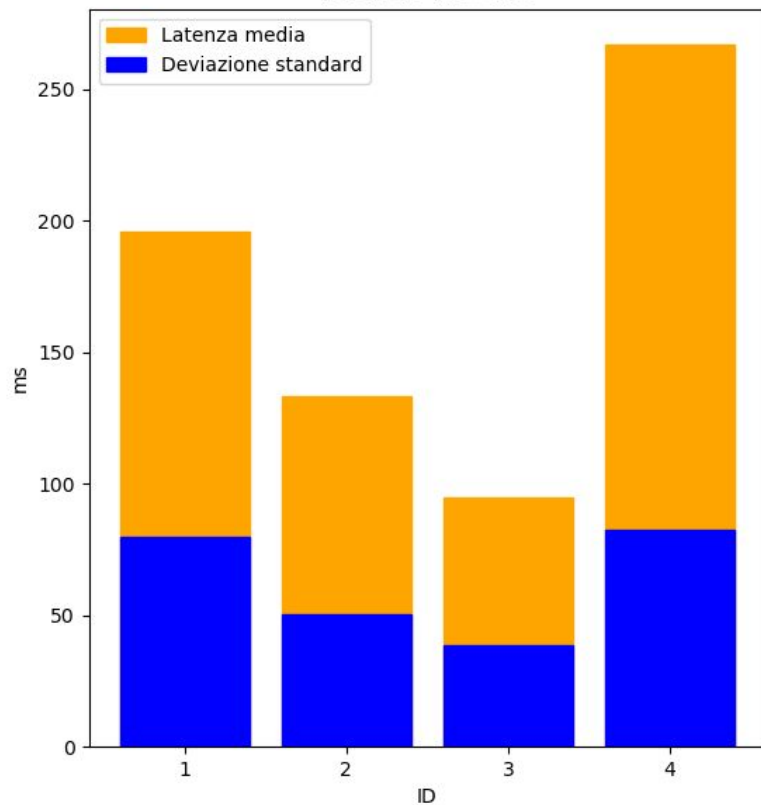
--- TEST IPFS BIG FILE---



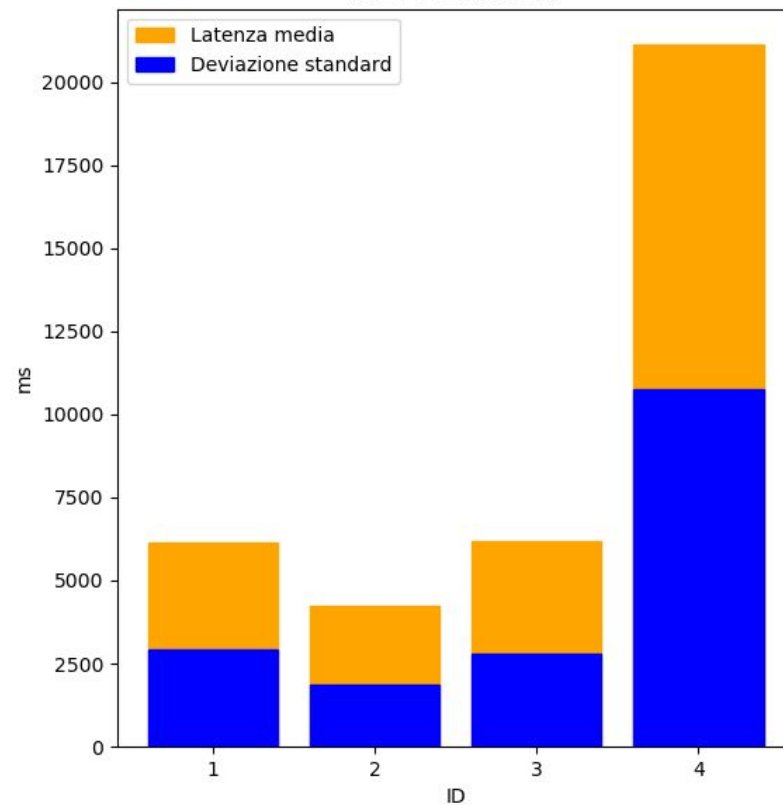
--- TEST SIA BIG FILE---



--- TEST BIG FILE IPFS ---



--- TEST BIG FILE SIA ---



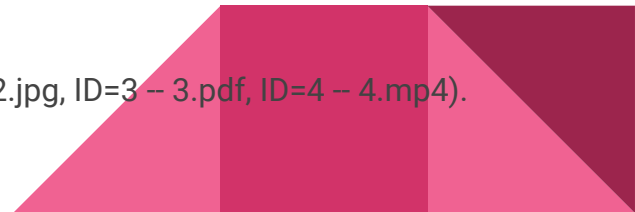
Grafici download dati

Di seguito verrà proposto il grafico relativo alle latenze ottenute durante il download dei 4 file presentati durante la fase di “caricamento big file”.

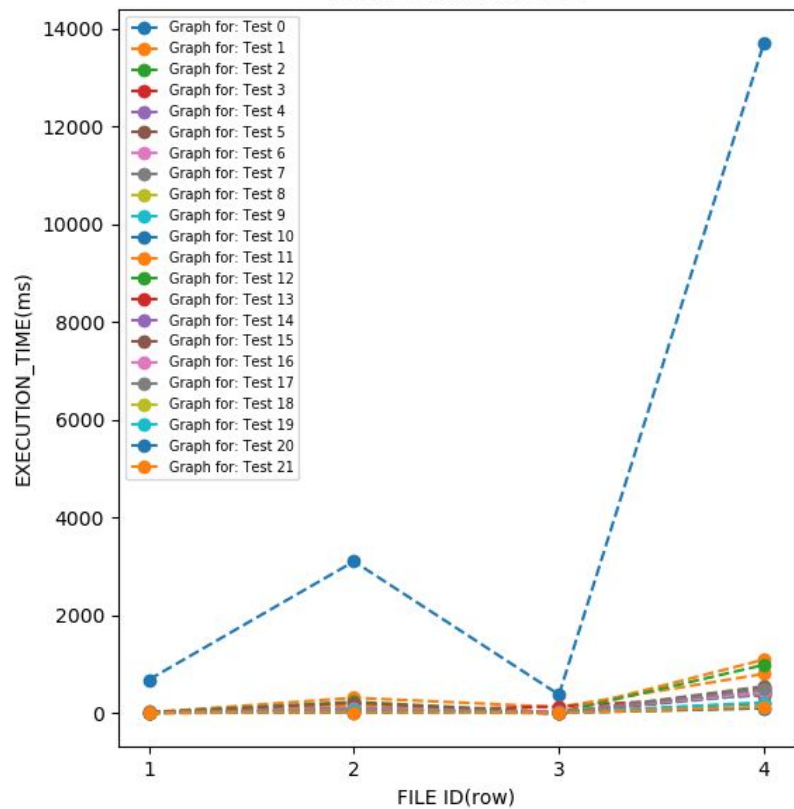
Sono proposti due grafici per ogni file:

- Delle latenze: rappresenta tutte le latenze ottenute per ogni test effettuato.
 - Asse X: id corrispondente ai file precedentemente presentati.
 - Asse Y: tempo di esecuzione, latenza in ms.
- Delle aggregazioni: sono rappresentate le informazioni circa latenza media, latenza accumulata e deviazione standard per ogni file.
 - Asse X: id relativo al file.
 - Asse Y: tempo di esecuzione, latenza in ms.

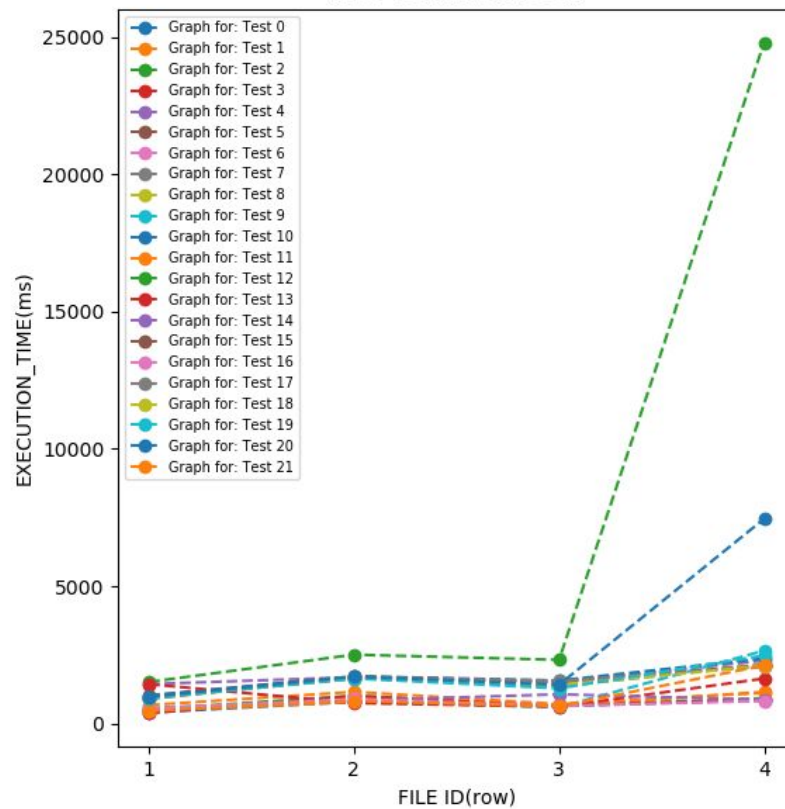
In tutti e due i grafici i file saranno rappresentati dai seguenti ID: (ID=1 -- 1.txt, ID=2 -- 2.jpg, ID=3 -- 3.pdf, ID=4 -- 4.mp4).



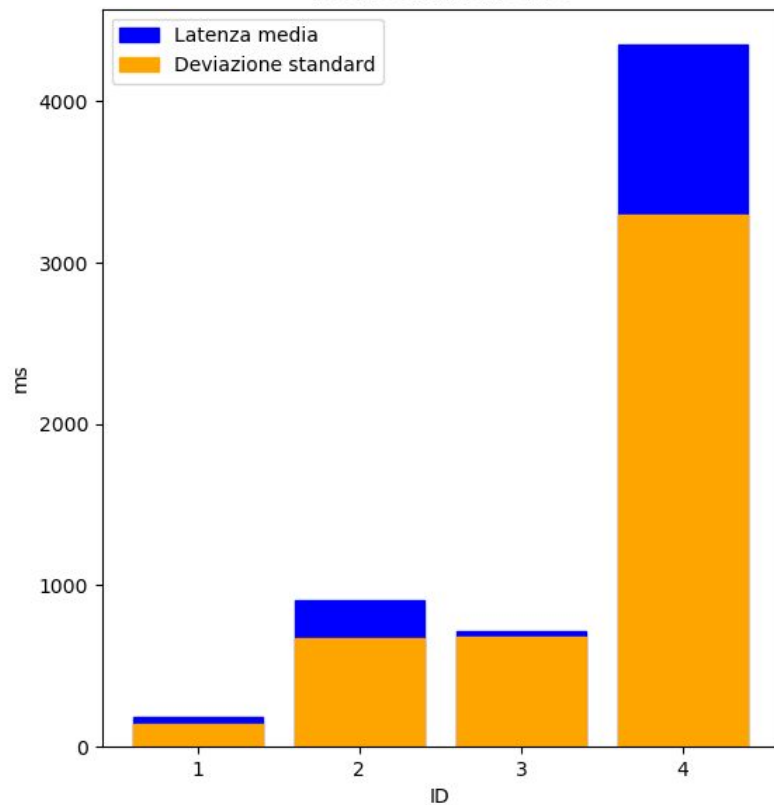
--- TEST DOWNLOAD IPFS ---



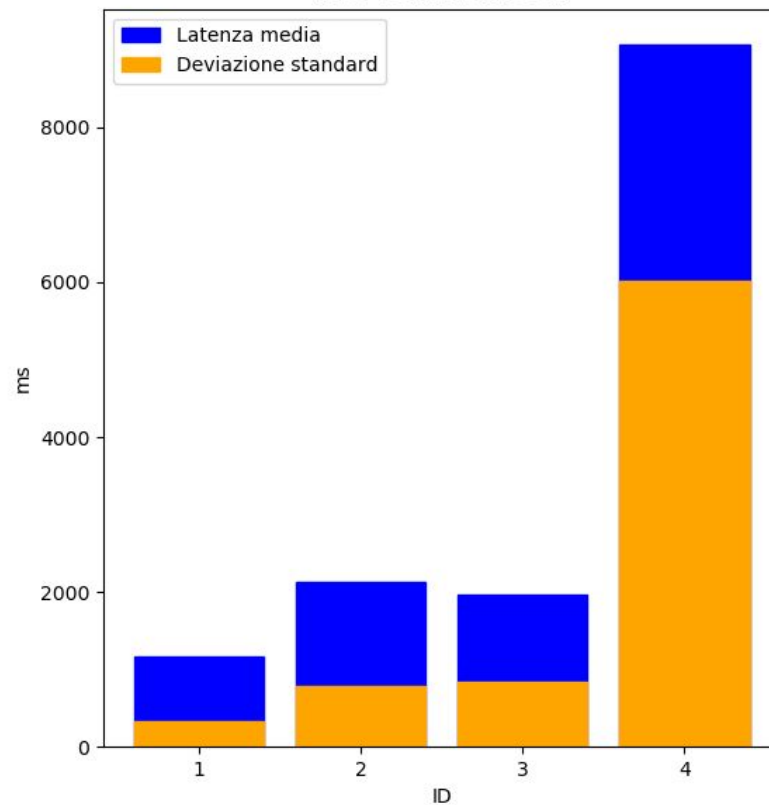
--- TEST DOWNLOAD SIA ---



--- TEST DOWNLOAD IPFS ---



--- TEST DOWNLOAD SIA ---



Conclusioni

I test effettuati utilizzando le due tecnologie hanno sottolineato delle latenze maggiori nell'utilizzo della testnet di Sia.

Queste sono dovute molto probabilmente al fatto che nel caso di caricamento e scaricamento dei file su rete IPFS le operazioni venivano effettuate partecipando come nodi attivi nella rete, mentre per quanto riguarda le operazioni effettuate su Skynet si faceva affidamento a un provider esterno e non si era partecipanti come nodi attivi sulla rete, comportando quindi latenze maggiori.

Altro fattore da tenere in considerazione è sicuramente il numero di nodi attivi su rete IPFS, piuttosto che su rete Skynet, infatti la rete IPFS è socialmente più utilizzata e conosciuta, rendendo quindi le operazioni all'interno di questa più veloci data la partecipazione di un maggior numero di nodi.



Bibliografia e sitografia

- Benet J., “IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)”, 14 Luglio 2014
<https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf>
- Ayush K., “integrazione IPFS”, 1 Maggio 2019
<https://www.dappros.com/it/201905/introduzione-a-ipfs/>
- Addaquay K., “A beginner’s guide to IPFS”, 21 Marzo 2018
<https://hackernoon.com/a-beginners-guide-to-ipfs-20673fedd3f>
- “IPFS implementation in Javascript”
<https://github.com/ipfs/js-ipfs>
- Champine L., Vorick D., “Sia: Simple Decentralized Storage”, 24 Novembre 2014
https://siasky.net/XABvi7JtJbQSMACDwnUnmp2FKDPjg8_tTTFP4BwMSxVdEg
- Vorick D., “Skynet”, 18 Febbraio 2020
<https://blog.sia.tech/skynet-bdf0209d6d34>
- “You Could Have Invented Skynet”
https://siasky.net/KABaBLrJzYkQ5GYWvVpbw5zRhUnQL_Onv2JU4Y8-arC8Vw

Tutto il materiale prodotto prende spunto dallo studio di ricerca:

Zichichi M., Ferretti S., D'Angelo G., “Are Distributed Ledger Technologies Ready for Smart Transportation Systems?”, Gennaio 2020