Minerva: Decentralized Collaborative Query Processing over InterPlanetary File System

Bowen Ding
Fudan University
bwding17@fudan.edu.cn

Yuedong Xu
Fudan University
ydxu@fudan.edu.cn

1 INTRODUCTION

The past decade has witnessed a more and more centralized Internet ecosystem. Content, usually generated by users, is stored and distributed by a small amount of data centers. As reported by Cisco, by 2021, only 0.96% of total Internet traffic will not originate from or be targeted at data centers[3]. Although managing data centrally has its advantages, it raises serious concerns that the data can be abused, and the equity of content owners can be infringed. A recent attempt to make content sharing distributed is InterPlanetary File System (IPFS), a protocol and network designed to create a content-addressable and peer-to-peer file system [1]. IPFS connects all computing devices with the same system of files, and allows users to hold a portion of the overall data for resilient storage. Despite the splendid promise of IPFS, it lacks sufficient applications beyond storage and sharing, where the distributed query service becomes an imperative.

Our purpose is to equip IPFS with flexible SQL query of data stored in a distributed network. To retrieve information contained in a dataset, one has to fetch the whole content first, and then perform the query locally. Three disadvantages emerge: i) unnecessary delay caused by downloading unnecessary data, if only part of the content contains the desired information; ii) need for extensive storage and computation resources, if the dataset is huge; and iii) inconvenience when redistributing updated data. This calls for the design of resilient distributed query with the goals of fine-grained query processing whose operations are performed collaboratively among distributed IPFS nodes.

We present Minerva, a decentralized data processing system over IPFS. It has these advantages:

- serving, retrieval, update and distribution of datasets is done in a decentralized fashion, with no single point of failure and low cost of storage;
- users who have the same dataset can accelerate the processing by collaboratively sharing computation resources;
- content addressing enables fine-grained access to a particular partition which the user is interested in, of the dataset.

There are a few related works that address similar problems. Authors in [2] focuses on version control and application integration in centralized data storage. A reliable data storage system that supports both cluster and P2P modes is presented in [7]. Google BigQuery [6] is a cloud-based Big Data-as-a-Service platform, featuring a huge collection of various open datasets.

2 SYSTEM ARCHITECTURE

Outlined in Figure 1, Minerva is a data query engine built upon a combination of three co-operating components: Qri[5] - the data management tool, Drill[4] - the distributed query engine, and IPFS[1] - the storage backend. Minerva takes advantages of the flexibility of Drill's query engine and the scalability of IPFS's decentralized filesystem. Each user of Minerva runs a local node, and the Minerva nodes form a peer-to-peer network. A user who has a particular dataset in her local store is a provider for that dataset, and can accept queries about it from other users in the network.

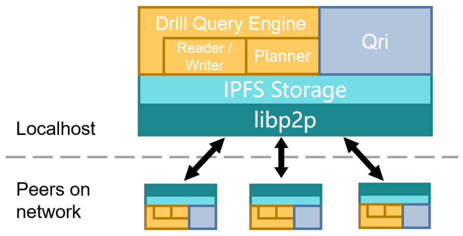


Figure 1: Minerva system overview.

Figure 2 illustrates how datasets are stored on IPFS to work with Minerva. With Qri, a dataset is first sliced into a number of chunks and each chunk becomes an object on IPFS. These pieces of data construct a hierarchical tree structure, where all leaf nodes contain the data, and the others record hashes of lower-level nodes, like directories. Each of the intermediate nodes and the root node have its own content identifier (CID), i.e. the hash of the content, and the CIDs serve as the paths to a certain part of or the whole dataset on IPFS, respectively. Users can input standard SQL statements into Drill, specifying as the table name the IPFS path of the part of data they want to query. The query string will look like the following:

```
SELECT `id`, `name`
FROM ipfs.`/ipfs/QmRhDW...3SVi/employees.json`
LIMIT 100
```

After parsing the SQL, Drill builds a distributed execution plan that will be sent to and executed on other nodes of Minerva who are serving the same dataset in the network,

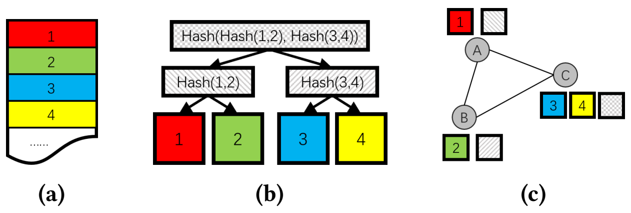


Figure 2: Minerva data layout: (a) data split into pieces, (b) tree structure holding data and (c) objects distributed among IPFS nodes.

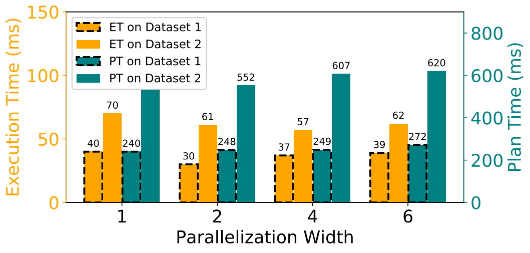
according to the DHT of IPFS. Drill is aware of data locality and tries to minimize the network cost by delegating computing jobs to the nodes that store particular pieces of data locally. Both decentralized ***read*** and ***write*** operations are enabled. Furthermore, Drill supports custom SQL functions, which are loaded from .jar files at runtime. Users can implement their conversion rules and analysis algorithms in the form of custom functions for a dataset, and distribute them with the dataset. Future users can benefit instantly by loading the custom functions from IPFS, in the same way as they specify a dataset stored on IPFS.
Demonstration: We have built an online service powered by a prototype system of Minerva to demonstrate its capabilities, available at http://www.datahub.pub/en.

3 PERFORMANCE EVALUATION

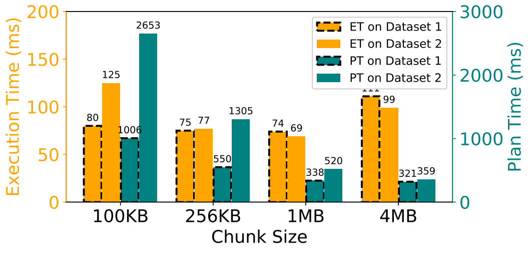
We have conducted preliminary performance evaluation on a prototype system in a 6-node cluster, each node running an instance of Minerva and IPFS working in private network mode. All statistics are averaged across 10 runs.

Figure 3a shows how parallelization width affects query performance. The chunk size is fixed at 1MB. For both datasets, the plan time sees a slight increase when the queries are executed on more nodes parallely, while the execution time first decreases then increases. It makes sense that the planner would need more time to gather enough information about more providers capable of handling queries. The case for execution time can be explained as a mixed effect of two factors: when the queries are less distributed, the overall execution time mainly depends on the slowest node; when they are executed on more nodes, the overhead, e.g. increased network communication costs that build up system load, becomes prominent.

We compare the impact of different chunk sizes on the performance in Figure 3b. The max parallelization width is set to 3 in this experiment. The chunk size has a significant impact largely on the plan time, which is the time spent on finding which node is the most suitable to execute a specific fragment of the query. This is because the smaller the chunks are, the more of them a dataset must be divided into, therefore there are more units of data that the scheduler will have to consider.



(a)



(b)

Figure 3: Query completion time under (a) different parallelization widths, (b) different chunk sizes. Dataset 1: 67MB, Dataset 2: 190MB.

4 CONCLUSIONS AND FUTURE WORK

We have presented Minerva, a decentralized query engine built upon IPFS that allows for collaborative and flexible query processing. We plan to further optimize it for better performance, in particular, the query planning phase that consumes most of the time. The current implementation of query planning is done locally. We believe a distributed refactoring will bring significant improvement in performance.

REFERENCES

[1] Juan Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. *CoRR* (2014). arXiv:1407.3561

[2] Anant Bhardwaj, Amol Deshpande, Aaron J. Elmore, David Karger, Sam Madden, Aditya Parameswaran, Harihar Subramanyam, Eugene Wu, and Rebecca Zhang. 2015. Collaborative Data Analytics with DataHub. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1916–1919.

[3] Cisco. 2019. Cisco Global Cloud Index: Forecast and Methodology, 2016-2021 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html. (2019). [Online; accessed 20-May-2019].

[4] Michael Hausenblas and Jacques Nadeau. 2013. Apache Drill: Interactive Ad-Hoc Analysis at Scale. *Big Data* 1, 2 (2013), 100–104. PMID: 27442064.

[5] qri.io. 2019. qri. https://github.com/qri-io/qri. (2019). [Online; accessed 20-May-2019].

[6] Kazunori Sato. 2012. An Inside Look at Google BigQuery. https://cloud.google.com/files/BigQueryTechnicalWP.pdf. (2012). [Online; accessed 20-May-2019].

[7] N. E. Taylor and Z. G. Ives. 2010. Reliable storage and querying for collaborative data sharing systems. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 40–51.