Chipmunk: Distributed Object Storage for NDN

Yong Yoon Shin, Sae Hyong Park, Namseok Ko, Arm Jeong
ETRI, GurumNetworks
Republic of Korea
{uni2u,labry,nsko}@etri.re.kr,{kjwonmail}@gmail.com

ABSTRACT

This demo shows an implementation of distributed object storage over NDN. It demonstrates how to reliably store and distribute data generated from sensors that do not have storage capability or generate large amounts of data. Chipmunk is an NDN based object storage system that leverages metadata that has data names and data locations to enable object storage service in a distributed system.

CCS CONCEPTS

• Information systems → Cloud based storage; **Distributed storage**; **Network attached storage**; **• Networks → Network File System (NFS) protocol**.

KEYWORDS

NDN, repository, file system, distribution, object storage, object based file system

ACM Reference Format:
Yong Yoon Shin, Sae Hyong Park, Namseok Ko, Arm Jeong. 2020. Chipmunk: Distributed Object Storage for NDN. In *ACM Conference on Information-Centric Networking (ICN '20), September 29–October 1, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3405656.3420231

1 INTRODUCTION

In this demo, we propose an enhanced object storage for NDN, called Chipmunk. Chipmunk is a distributed object storage that is enhanced from repo-ng [3]. In Chipmunk, data is stored in the vicinity of producers and its metadata is put in another node according to the hash value of its name. The metadata has the information of how to access the data including the data name. Consumers can easily find the information on data from the metadata which can be found first using the hash value of the data name.

2 DESIGN

We first present an overview of Chipmunk's architecture and discuss the design details.

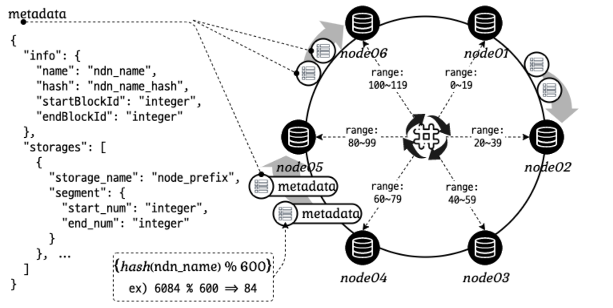
2.1 Overview

Object storage [2] uses unique identifiers to map objects to metadata as a part of data management. Chipmunk is a distributed object storage system for NDN, in which the identifiers are the hash values of data names. Figure 2 shows the overview of Chipmunk architecture. Every node in Chipmunk has its own name and decides the range of data identifiers that it is responsible for based on the hash value of its name. Chipmunk nodes are configured with the same common prefix, and each node is separated by a node ID. As shown in Figure 1, all nodes use the same 'common-prefix', so the user does not need to know the complete prefix of the node. A data producer splits the data into segments, if configured so. Each node in Chipmunk has two types of storage for data and metadata: file storage and metadata store. Data segments are stored in the file system of requested node and its metadata is stored in the node decided by the data identifier (hash value).

```
/<common-name>/<command-verb>/.................../......................
 common-prefix  request-command  RepoCommandParameter  Signed Interest Components

   USER          TOOL
----------   -----------
[provider]   $ndnputfile  <command>  common-prefix  ndn-name  filename
[consumer]   $ndngetfile  <command>  ndn-name
```

Figure 1: Chipmunk commands and tools

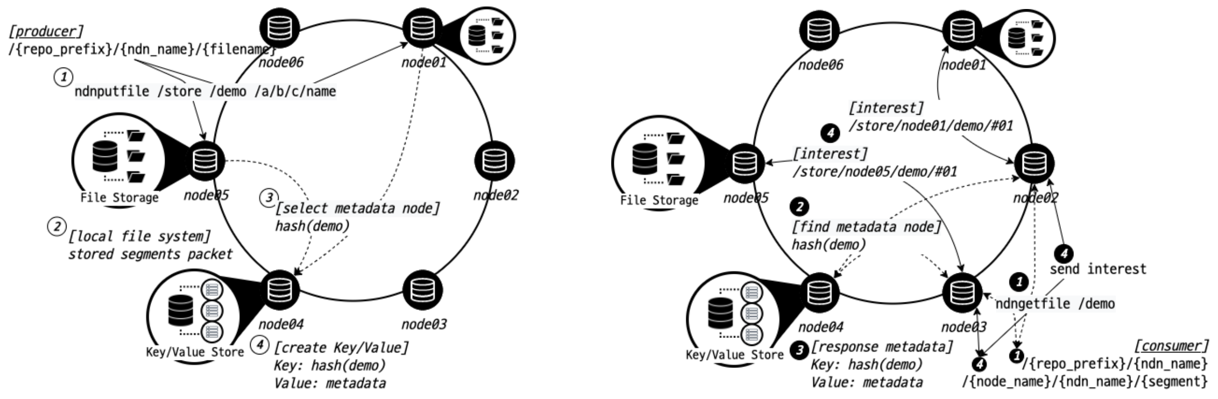


```
{
  "info": {
    "name": "ndn_name",
    "hash": "ndn_name_hash",
    "startBlockId": "integer",
    "endBlockId": "integer"
  },
  "storages": [
    {
      "storage_name": "node_prefix",
      "segment": {
        "start_num": "integer",
        "end_num": "integer"
      }
    }, ...
  ]
}
```

Figure 2: The Architecture Overview of Chipmunk

2.2 File Storage

The file storage of Chipmunk is based on an ext4 file system, which is a default file system in Linux. Data producers request data insertion to a Chipmunk node using signed Interest with 'insert' command. The command is accompanied with RepoCommandParameter which includes the name of data ('ndn_name') and segmentation information. The requested Chipmunk node retrieves the data (segments) from the data producer using the 'ndn_name.' Each data (segment) is stored in a directory whose name is the first 2-byte of hash value of the data segment name. While the data (segment) is being stored, its name is prefixed by node's ID such as '/node-name.' After storing all the data segments in the file storage,



(a) Data Insertion **(b) Data Retrieval**

Figure 3: The Demo Overview of Chipmunk

its metadata is created and stored in the node decided by the hash value of 'ndn_name.'

2.3 Metadata Store

The metadata store is a key/value store providing metadata with the hash value of 'ndn_name' as a key. As in Figure 2, based on the result of the hashing of its node name, a node takes care of the 'ndn_name' that are closest in Euclidean distance [1] and stores its metadata. The data consumer sends the request to 'common-prefix' and 'ndn_name' because the prefix of the node where the data is stored is unknown, and obtains metadata based on the hash value of 'ndn_name' regardless of the network location. Metadata includes the information such as 'ndn_name', hash values for 'ndn_name', information about Chipmunk nodes that store the data, and segment numbers.

3 SHOWCASE

Figure 3 shows a high-level overview of Chipmunk based on a scenario consisting of several nodes.

3.1 Data Insertion

As shown in Figure 3a, the data segments are stored in the following order:

① The producer requests to insert a data segment.
- ndnputfile <common-prefix> <ndn_name> <filename>

② A Chipmunk node, which receives an insert request, pulls the data through the exchange of Interest and Data messages, and store it in the file system.
Note that the same data can be requested to be stored in multiple Chipmunk nodes for some purposes such as increasing resiliency.

③ A metadata is created and the decision about selecting a node to store the metadata is made by a hash calculation on ***ndn_name***.

④ The request to store the metadata is delivered to the selected metadata store node.

3.2 Data Retrieval

As shown in Figure 3b, the retrieval procedure of a data is performed as follows, regardless of the location of consumers:

❶ The consumer requests to get a data with ***ndn_name***.
- ndngetfile <ndn_name>

❷ The node that gets the request from the consumer, which is usually the closest to the consumer, find the metadata for the data.

❸ The metadata is returned to the consumer. The metadata has the information where the data is stored.

❹ The consumer sends a request to a node that actually stores the data. When there are multiple nodes storing the data, consumers can choose one of them depending on the policies.
- /<Chipmunk-node-prefix>/<ndn_name>/<segment_num>

4 FUTURE WORK

In this demo, we showed the design of Chipmunk which is a distributed object storage for NDN. The key idea of Chipmunk is to bring in the concept of Object Storage into the NDN realm to satisfy the increasing demands for Object Storage. As the next step, we are planning to implement a Merkle-tree [4] mechanism into Chipmunk, and conduct benchmark tests using large files. Also, we hope to share our source code with NDN communities and actively get feedback through various channels.

ACKNOWLEDGMENTS

This work was supported by the ICT R&D program of MSICT/IITP. [2017-0-00045, Hyper-connected Intelligent Infrastructure Technology Development]

REFERENCES

[1] Per-Erik Danielsson. 1980. Euclidean distance mapping. *Computer Graphics and image processing* 14, 3 (1980), 227–248.
[2] Kristal T Pollack and Scott A Brandt. 2005. Efficient access control for distributed hierarchical file systems. In *22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*. IEEE, 253–260.
[3] repo-ng contributors. 2014. repo-ng: Next generation NDN repository. https://redmine.named-data.net/projects/Repo-ng/wiki
[4] Yong Yoon Shin and Sae Hyong Park. 2019. Canary: a Scalable Content Integrity Verifying Protocol for ICN. In *Proceedings of the 6th ACM-ICN*. 167–168.