Петров Д.М. КИТ-М120Б

Лабораторна робота № 5

Геш дерева

Мета: Ознайомитись з технологією MerkleTree

Завдання: Створити екосистему (та продемонструвати її роботу), що складається з наступних компонентів:

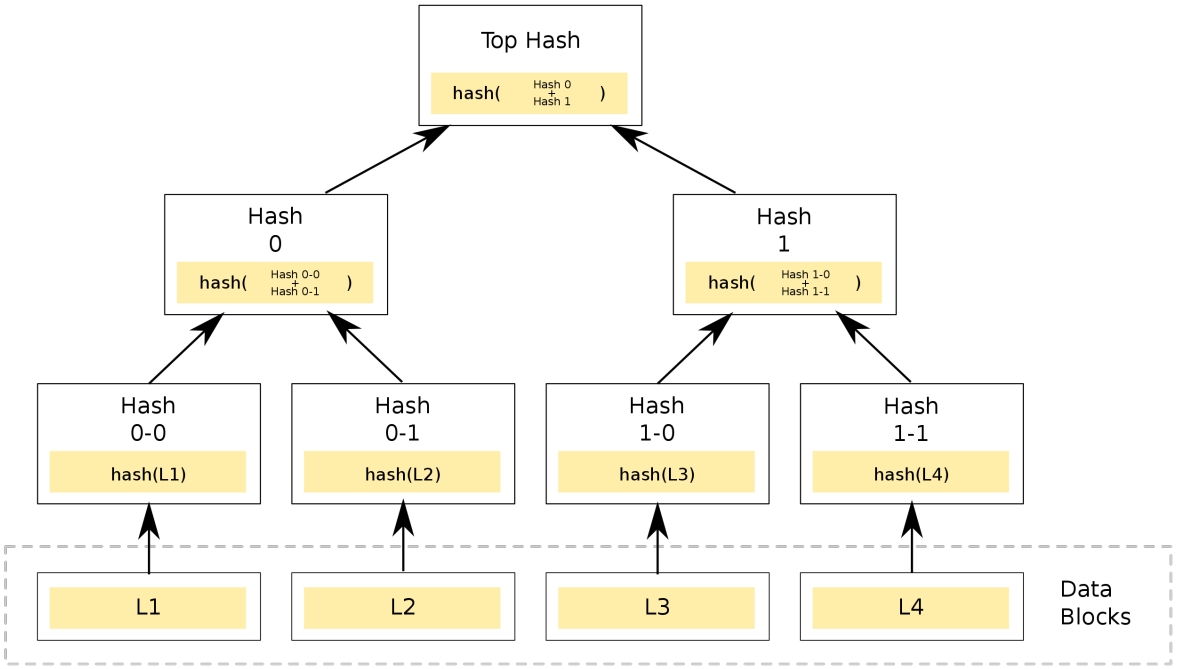
- сервер. Має інформацію о файлах мережі у наступному вигляді:
  - merkle root
  - Перелік клієнтів, що мають хоча б частину контенту файлу та перелік блоків, які вони мають
- клієнт, що має хоча б частину контенту файлу. Дії, що можна проводити над клієнтом:
  - запитати частину файла. На вхід подається merkle root цього файлу, індекс блоку. На вихід дається контент файлу або помилка, якщо такого блоку немає
  - виконати верифікацію блока. На вхід подається merkle root, block hash. На вихід подається обрізане під-дерево (гілку до запитаного геша) стосовно алгоритму. Якщо block hash відсутній або невалідний - повертаєтсья стосовна помилка.
- клієнт, що завантажує файл. Послідовність дій, що виконується:
  - отримує будь-яким чином merkle root бажаного для завантажування файлу
  - виконує завантаження блоку:
    - питає у сервера перелік клієнтів, що має певну частину цього файлу.
    - завантажує блок
  - виконує верифікацію блоку:
    - виконує гешування блоку
    - обирає будь-який інший сервер, що має цей блок та питає його частину merkle дерева. Якщо інший сервером нема, питаємо у того, з якого завантажували
    - самостійно проводить верифікацію гілки дерева
    - якщо усе добре - зберігає блок, оновлює внутрішню базу даних, посилає запит на сервер для додання зпису, що даний клієт має блок з тиким індексом для певного файлу (merkle root)

Хід роботи

Дерево Меркла (геш-дерево, tiger tree tashing, англ. *Merkle tree*) представляє собою особливу структуру даних, яка містить підсумкову інформацію про якийсь більший обсяг даних. Використовується для перевірки цілісності даних.

Дані поділяються на малі частини (блоки), які індивідуально гешуються за допомогою Leaf Tiger Hash, потім з кожної пари гешів по черзі обчислюється Internal Tiger Hash. Якщо до гешу немає пари, то він переноситься в новий ланцюжок без змін. Далі в ланцюжку для кожної пари знову обчислюється Internal Tiger Hash. Ця процедура повторюється до тих пір, поки не залишиться один геш. Цей один геш, що залишився, називають Tiger Tree Root. Саме його використовують для однозначної ідентифікації файлу і вказують у різних P2P посиланнях.

```
Level          Tiger Tree Root
|                    /
0:              --- 21 -- -------\
               /          \        \
1:         - 20 -           19 ------\
          /      \          \        \
2:      17         18         19 ------ Internal Tiger Hashes
      /   \       /   \      /   \     /
3:   12    13    14    15   16   11 --/
     /\    /\    /\    /\   /\    \
4: 1  2 3  4 5  6 7  8 9 10  11 ---  Leaf Tiger Hashes
```



Геш-дерева використовуються для перевірки даних, що зберігаються, обробляються та передаються в комп'ютерах та між ними. Забезпечують перевірку на цілісність та достовірність даних та окремих блоків, що передаються між вузлами peer-to-peer мережі. Геш-дерева застосовуються в системах trusted computing. Геш-дерева також використовуються в геш-криптографії.

Геш-дерева використовуються у файлових системах IPFS, Btrfs та ZFS для протидії Деградації даних, програмі розповсюдженя даних Dat, протоколі Google Wave, розподіленій системи керування версіями Git та Mercurial, резервній системі Tahoe-LAFS , однорангових мережах Bitcoin та Ethereum, фреймворку прозорості сертифікатів, системах Riak та Dynamo.

Початкова реалізація дерева Меркла у Bitcoin від Сатосі Накамото застосовує крок стиснення геш-функції до надмірного рівня, що полегшує використання дерев типу Fast Merkle.

Було розроблено дві програми, клієнт та сервер.

Сервер, як описано в завданні, зберігає геш значення merkle root файлу, для того, щоб була можливість перевірити який файл потребує клієнт. Сервер зберігає інформацію о клієнтах, які мають хоча б якусь частину файлу.

Клієнт зберігає в пам’яті частину файлу та запитує у сервера інформацію о клієнтах які мають частини файлів яких в нього немає. Отримавши інформацію, клієнт запитує у клієнта частину файлу та одночасно приймає запити на відправку частин файлу які в нього вже є.

Сервер:

```
Open
Connection: 127.0.0.1:49903
Set/All/10102/????Y???Q??&3?
Connection: 127.0.0.1:49904
File/message.txt
Connection: 127.0.0.1:49905
Port/0/????Y???Q??&3?
Port for client = 10102
Port/10102
Connection: 127.0.0.1:49908
Set/0/10104/????Y???Q??&3?
Connection: 127.0.0.1:49909
Port/1/????Y???Q??&3?
Port for client = 10102
Port/10102
Connection: 127.0.0.1:49912
Set/1/10104/????Y???Q??&3?
Connection: 127.0.0.1:49913
Port/2/????Y???Q??&3?
Port for client = 10102
Port/10102
Connection: 127.0.0.1:49916
Set/2/10104/????Y???Q??&3?
Connection: 127.0.0.1:49917
Port/3/????Y???Q??&3?
Port for client = 10102
Port/10102
Connection: 127.0.0.1:49920
Set/3/10104/????Y???Q??&3?
^CSign Int
```

Клієнт, що має хоча б якусь частину файлу:

```
Size read from file = 20480
Size read from file = 20480
Size read from file = 20480
Size read from file = 14952
########## Port ############
10101
Was opened port = 10102
Size empty blocks = 0
Size fill blocks = 4
Set/All/10102/????Y???Q??&3?
Check
Send blocks
Check
#################################
INDEX = 3
#if#
Test 1
F = 3
S = 4
F = 1
S = 2
Response = Val/2
Check
Send blocks
Check
#################################
INDEX = 3
INDEX = 4
#if#
Test 1
F = 4
S = 3
F = 1
S = 2
Response = Val/2
Check
Send blocks
Check
#################################
```

Клієнт, що запитує частину файлу:

```
File with settings Empty.
Request was send.
########## Port ############
10103
Was opened port = 10104
Size empty blocks = 4
Size fill blocks = 0
Port server on client = 10104
Port/0/????Y???Q??&3?
Port/10102
Port client with block = 10102
Get/0/????Y???Q??&3?
Recv block with size = 20480
Val/????Y???Q??&3?/-???o??
                         ?2    !???

Val/2
sizeString = 2
S = 4
S = 2
SIZE = 2
F = 3
0x2d 0xc6 0xf8 0xf9 0x6f 0x3f 0xdb 0x0c 0xe4 0x32 0x09 0x21 0x83 0xf9 0x8a 0x0c
S = 4
0x7f 0x2b 0xe6 0x76 0x60 0x3e 0x17 0x03 0x8d 0x5a 0x47 0x39 0x8a 0x1a 0x81 0xea
F < S
###############
INDEX = 1
0x57 0x05 0x6b 0xc9 0x51 0x75 0xb8 0x49 0x90 0x10 0x54 0x30 0xcc 0xfb 0x38 0x04
F = 1
0x57 0x05 0x6b 0xc9 0x51 0x75 0xb8 0x49 0x90 0x10 0x54 0x30 0xcc 0xfb 0x38 0x04
S = 2
0x6a 0x88 0x29 0x6f 0x94 0x30 0x83 0xd5 0x3a 0xca 0xb2 0x36 0xe0 0x68 0x00 0x10
F < S
###############
INDEX = 0
0x81 0xc3 0xc9 0xd8 0x59 0xc0 0xd0 0x0e 0xc2 0x51 0xbd 0xd9 0x26 0x33 0xee 0x9a
Validate Ok
Set/0/10104/????Y???Q??&3?
```

Висновок:

Ознайомився з технологією MerkleTree. Створив екосистему (та продемонстрував її роботу), що складається з наступних компонентів: сервера та клієнтів.