실험용 stateDB 세팅 과제

실험용 stateDB 세팅

프로젝트 결과물의 성능을 측정하기 위해, 미리 세팅(계정 및 컨트랙트가 생성)된 stateDB가 필요함

현재 레포지토리의 storage 디렉토리 속 코드가 현재 이를 담당하고 있음

Goal: 로컬 트랜잭션 및 크로스-샤드 트랜잭션 용을 구분하여 컨트랙트 계정 stateDB에 세팅

이전 과제 결과물까지의 실험용 stateDB 생성 과정

1. docker를 통해 state shard 작동 시, storage/create_storage.go 속 main 함수 호출
2. n개의 랜덤 이더리움 주소를 결정론적(항상 동일한 값)으로 생성 후, txt 파일로 저장
3. 각 m개의 Travel, Hote, Train 컨트랙트 이더리움 주소를 랜덤하되 결정론적으로 생성 후, txt 파일로 저장
4. 계정 주소 txt 파일 속 주소를 하나씩 읽어가며 해당하는 state shard stateDB에 계정 생성 with 1000ETH

> 💡 현재 계정의 소속 샤드를 구별하는 방법은 다음과 같음.
>
> 계정 주소를 16진수로 치환한 것의 마지막 바이트 값을 전체 state shard 수로 나눈 것의 나머지(`accountAddressInByte[-1] % NumShard`)

4. 컨트랙트 주소 txt 파일 속 주소를 하나씩 읽어가며 해당하는 컨트랙트 코드로 해당하는 state shard stateDB에 컨트랙트 계정 생성
5. 각 state shard의 stateDB를 실제 파일로써 디스크에 저장(stateDB.Commit 그리고 tdb.Commit)

ToDo

- Travel 컨트랙트와 연결되는 Train 그리고 Hotel 컨트랙트를 연결하는 현재 방식은 다음과 같음
  - 랜덤으로 생성된 동일한 수의 Travel, Train, Hotel 컨트랙트 주소 파일 중, 특정 Travel 컨트랙트 주소와 동일한 index에 위치한 Train 그리고 Hotel 컨트랙트 주소 선택
  - e.g., 5번째 줄에 저장된 Travel 컨트랙트 주소는 5번째 줄에 저장된 Train 컨트랙트 주소와 5번째 줄에 저장된 Hotel 컨트랙트 주소와 연결
- 위 경우, 컨트랙트 주소는 랜덤으로 생성되기에 임의로 선택한 Travel 컨트랙트 주소가 로컬 트랜잭션을 발생시키는지 크로스-샤드 트랜잭션을 발생시키는지는 비결정적
  - 아마 이것이 비결정적인 컨트랙트 호출 성공/실패의 원인
- 이와 달리 실제 실험에 사용할 워크로드를 생성 시, 로컬과 크로스-샤드를 구분하여 생성해야 함
- 최종 목표는 실험자가 특정 Travel 컨트랙트 주소가 로컬 트랜잭션을 발생시키는지 크로스-샤드 트랜잭션을 발생시키는지 구분할 수 있도록 하는 것
  - 이를 통해 실험자는 자유롭게 로컬/크로스-샤드 트랜잭션의 비율을 결정할 수 있음

Instruction

로컬 및 크로스-샤드 Travel 컨트랙트 호출 트랜잭션을 발생시키는 파이썬 스크립트 작성

1. 생성된 Travel, Train, Hotel 컨트랙트 주소 파일을 확인하여 특정 Travel 컨트랙트 주소가 어떤 state shard에서 호출되는 것인지 그리고 로컬 또는 크로스-샤드 트랜잭션을 발생시키는지 주소인지 체크하고 이를 전역 변수로 저장

```
# global variable for distinguishing Travel contract address # example
using array, maybe there are better ways state_shard_1_local_travel =
[] state_shard_1_cross_travel = [] state_shard_2_local_travel = []
state_shard_2_cross_travel = [] ... state_shard_6_local_travel = []
state_shard_6_cross_travel = [] # distinguish Travel CA # whether it is
local or cross, and its deployed state shard for i, travel_addr in
enumerate(all_travel_addrs): train_addr_hex = all_train_addrs[i][2:]
hotel_addr_hex = all_hotel_addrs[i][2:] travel_shard_id =
travel_addr[-1] % numShards train_shard_id = all_train_addrs[i][-1] %
numShards hotel_shard_id = all_hotel_addrs[i][-1] % numShards
is_cross_shard = (travel_shard_id == train_shard_id == hotel_shard_id)
# add Travel CA to global array using travel_shard_id and
is_cross_shard ...
```

2. 그리고 해당 전역 변수를 사용하여 실험자가 타켓팅(state shard, local 또는 cross)한 트랜잭션을 state shard로 전파하여 테스트하는 함수 작성 후 테스트

```
# function that create targeted tx # and send that tx to corresponding
state shard using /tx/submit rpc def send_targeted_tx(shard_ID,
is_cross): ...
```