연구 보고서

작성자	김한호	작성일자	2021.05.16.

1. 연구 계획

- e2-list-private-data 프로젝트 체인코드 함수 작성 테스트코드 작성
- 성능측정 논문 검색 정리 적용 고민

2. 논문 연구 진행

```
[
 {
   "name": "civilCollection",
   "policy": "OR('civilMSP.member', 'controlMSP.member')",
   "requiredPeerCount": 0,
   "maxPeerCount": 3,
   "blockToLive":1000000,
   "memberOnlyRead": true,
   "memberOnlyWrite": true
 },
 {
   "name": "civilPrivateCollection",
   "policy": "OR('civilMSP.member')",
   "requiredPeerCount": 0,
   "maxPeerCount": 1,
   "blockToLive":3,
   "memberOnlyRead": true,
   "memberOnlyWrite": false,
   "endorsementPolicy": {
     "signaturePolicy": "OR('civilMSP.member')"
   }
 },
 {
   "name": "controlPrivateCollection",
   "policy": "OR('controlMSP.member')",
   "requiredPeerCount": 0,
   "maxPeerCount": 1,
   "blockToLive":3,
   "memberOnlyRead": true,
   "memberOnlyWrite": false,
   "endorsementPolicy": {
     "signaturePolicy": "OR('controlMSP.member')"
   }
 }
]
~
~
~
~
~
~
~
~
~
~
~
"collections_config.json" 35L, 806C                    1,1           모두
```

collections_config.json을 통해 쓰기/읽기 제한을 하고 접근 제한을 하기 위한 여러 정책을 작성함. 간단한 구현을 한 이후 확대 적용을 위해 civil 조직과 control

조직간의 쓰고 읽기만을 구현하기로 함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go$ ls
META-INF  README.md  chaincode  collections_config.json  go.mod  go.sum  main.go  vendor
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go$ cd chaincode/
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go/chaincode$ ls
asset_queries.go       asset_transfer.go       e2_list_enter.go       mocks
asset_queries_test.go  asset_transfer_test.go  e2_list_enter_test.go
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go/chaincode$
```

접근제한을 하는 파일은 프로젝트 폴더 최상위에 위치시킴. 기존 복사해온 asset-transfer-private-data의 파일은 참조를 위해 chaincode 폴더내에 위치시킴. 새로 작성하는 파일은 e2_list_enter.go로 명명함. 테스트 파일은 e2_list_enter_test.go로 테스트파일은 복사해온 파일과 맞춰서 정함.

```
19
20 const civilCollection = "civilCollection"
21 const permittedAgreementObjectType = "permittedAgreement"
22
23 // Civil is distinguished by a random variabled ID
24 type Civil struct {
25     ID              string `json:"civilID"`
26 }
27
28 // CivilPrivateDetails describes details that personal information to civil
29 // Write Organization: Civil
30 // Read Organization: Civil
31 // Permitted Read: Control
32 type CivilPrivateDetails struct {
33     ID              string  `json:"civilID"`
34     Name            string  `json:"civilName"`
35     PhoneNumber     string  `json:"civilPhoneNumber"`
36     Address         string  `json:"civilAddress"`
37 }
38
39 // CivilInfectionStatus describe infection status in Civilian
40 // Write Organization: Control
41 // Read Organization: Control
42 // Permitted Read: Civil
43 // three states, 'normal', 'suspected', and 'infected'
44 type CivilInfectionStatus struct {
45     ID              string  `json:"civilID"`
46     Status          string  `json:"civilStatus"`
47 }
48
49 // ReqPrivacy is a struct by in which a person decides whether to allow personal information a
   t the request of an agency.
50 // Write Organization: Civil
51 // Read Organization: Civil, Control
52 type ResPrivacy struct {
53     ID              string  `json:"resprivID"`
54     IsPermitted     bool    `json:"isPermitted"`
55 }
56
57 // Control is a management agency to request and track the personal information of a person wi
   th an infectious disease. It consist of randomly assigned ID and organization names.
58 // Write Organization: Control
59 // Read Organization: Any
60 type Control struct {
61     ID              string  `json:"controlID"`
62     OrgName         string  `json:"controlName"`
63 }
                                                                63,1          6%
```

e2_list_enter.go에는 생각했던 변수를 정의함. 차후 구현 시 조직들의 읽기/쓰기 권한을 기억하기 위해 코멘트에 작성함. 나중 구현이 변하게 되면 다시 작성하여 시간을 줄일 수 있도록 함.

```
65 // ReqPrivacy is a data structure for requesting personal information. Identifier of the indiv
   idual requesting The identifier of the organization requesting The date of the requesting star
   t and end and permission are stored. Permission status is specificed as true by default when c
   reated, but this is implemented through a process that is subsequently implemented in chaincod
   e or inserted on-demand changed in DApp.
66 // Write Organization: Control
67 // Read Organization: Control, Civil
68 type ReqPrivacy struct {
69     ID              string  `json:"reqprivID"`
70     CivilID         string  `json:"civilID"`
71     ControlID       string  `json:"controlID"`
72     ReqStartDate    string  `json:"reqstartDate"`
73     ReqEndDate      string  `json:"reqendDate"`
74     IsPermitted     bool    `json:"isPermitted"`
75 }
76
77 // Shop is consist of four variables. first identifier of store. second name of store. third t
   elephone-number. fourth address of shop.
78 // Write Organization: Shop
79 // Read Organization: ALL
80 type Shop struct {
81     ID          string `json:"shopID"`
82     Name        string `json:"shopName"`
83     Telephone   string `json:"shopTelephone"`
84     Address     string `json:"shopAddress"`
85 }
86
87 // StoreVisitList are recorded when citizens visit the store visits the store. shopID civilID
   VisitTime saved.
88 // Write Organization: Civil, Control
89 // Read Organization: Control
90 type StoreVisitList struct {
91     ID          string  `json:"visitID"`
92     ShopID      string  `json:"shopID"`
93     CivilID     string  `json:"civilID"`
94     VisitTime   string  `json:"visitTime"`
95 }
96 // CreateCivil creates a new civil by placing the main info details in the CivilPrivateDetails
97 // that can be read by all organizations. pernoal information is stored in the civils org spec
   ific collection
98 func (s *SmartContract) CreateCivil(ctx contractapi.TransactionContextInterface) error {
99     // Get new civil from transient map
100    transientMap, err := ctx.GetStub().GetTransient()
101    if err != nil {
102        return fmt.Errorf("error getting transient: %v", err)
103    }
@
                                                                    103,1-4       23%
```

설계했던 생각과 다르게 개인정보를 요청한 기간과 승인시에 어떻게 기록을 남길지 고려함. 새롭게 요청하는 Control에서 개인정보를 요청하는 ReqPrivacy 구조를 정의한다. ResPrivacy를 통해 Civil 조직에 개인정보를 요청을 받아서 수락/거절 처리를 하는 스트럭트를 정의함. 프로그램 절차를 고민했을 때 asset-transfer 예제를 참조하기로 함. asset-transfer에서는 거래가(appraisedValue)를 통해 거래를 성립하는 것을 프로세스를 적용하는 것을 생각하기로 함.

```
 98 func (s *SmartContract) CreateCivil(ctx contractapi.TransactionContextInterface) error {
 99     // Get new civil from transient map
100     transientMap, err := ctx.GetStub().GetTransient()
101     if err != nil {
102         return fmt.Errorf("error getting transient: %v", err)
103     }
104     // Civil properties are private, therefore they get passed in transient field, instead of
    func args
105     transientCivilJSON, ok := transientMap["civil_properties"]
106     if !ok {
107         //log error to stdout
108         return fmt.Errorf("civil not found in the transient map input")
109     }
110
111     type civilTransientInput struct {
112         ID          string  `json:"civilID"`
113         Name        string  `json:"civilName"`
114         PhoneNumber string  `json:"civilPhoneNumber"`
115         Address     string  `json:"civilAddress"`
116         Status      string  `json:"civilStatus"`    //Type is used to distinguish the various
    type of civil in state
117     }
118
119     var civilInput civilTransientInput
120     err = json.Unmarshal(transientCivilJSON, &civilInput)
121     if err != nil {
122         return fmt.Errorf("failed to unmarshal JSON: %v", err)
123     }
124     if len(civilInput.ID) == 0 {
125         return fmt.Errorf("civilID field must be non-empty string")
126     }
127     if len(civilInput.Name) == 0 {
128         return fmt.Errorf("civilName field must be non-empty string")
129     }
130     if len(civilInput.PhoneNumber) == 0 {
131         return fmt.Errorf("civilPhoneNumber field must be non-empty string")
132     }
133     if len(civilInput.Address) == 0 {
134         return fmt.Errorf("civilAddress field must be non-empty string")
135     }
136     if len(civilInput.Status) == 0 {
137         return fmt.Errorf("Status field must be a non-empty string")
138     }
139     // check if civil already exists
140     civilAsBytes, err := ctx.GetStub().GetPrivateData(civilCollection, civilInput.ID)
141     if err != nil {
142         return fmt.Errorf("failed to get civil: %v", err)
                                                                    141,1-4       37%
```

기존 생성부터 했던 예제와 다르게 일시적인 변수를 생성해서 실제로 저장하는 과정을 거침. 코드에 들어간 다량의 조건문은 입력되어야 할 값이 제대로 입력되었는지 확인함.

```
139     // check if civil already exists
140     civilAsBytes, err := ctx.GetStub().GetPrivateData(civilCollection, civilInput.ID)
141     if err != nil {
142         return fmt.Errorf("failed to get civil: %v", err)
143     } else if civilAsBytes != nil {
144         fmt.Println("Civil already exists: " + civilInput.ID)
145         return fmt.Errorf("this civil already exists: " + civilInput.ID)
146     }
147     // Get ID of submitting client identity
148     clientID, err := submittingClientIdentity(ctx)
149     if err != nil {
150         return err
151     }
152
153     // Verify that the client is submitting request to peer in their organization
154     // This is to ensure that a client from another org doesn't attempt to read
155     // or write private data from this peer.
156     err = verifyClientOrgMatchesPeerOrg(ctx)
157     if err != nil {
158         return fmt.Errorf("CreateCivil cannot be performed: Error %v", err)
159     }
160
161     // Make submitting client the civil
162     civil := Civil{
163         ID:     civilInput.ID,
164     }
165     civilJSONasBytes, err := json.Marshal(civil)
166     if err != nil {
167         return fmt.Errorf("failed to marshal civil into JSON: %v", err)
168     }
169
170     // Save civil to private data collection
171     // Typical logger, logs to stdout/file in the fabric managed docker container, running thi
    s chaincode
172     // Look for container name like dev-peer0.civil.example.com-{chaincodename_version}-xyz
173     log.Printf("CreateCivil Put: collection: %v, ID %v, client %v", civilCollection, civilInpu
    t.ID, clientID)
174
175     err = ctx.GetStub().PutPrivateData(civilCollection, civilInput.ID, civilJSONasBytes)
176     if err != nil {
177         return fmt.Errorf("failed to put civil into private data collection: %v", err)
178     }
179
180     // Save civil personal information to collection visible to owning organization
181     civilPrivateDetails := CivilPrivateDetails{
182         ID:          civilInput.ID,
183         Name:        civilInput.Name,
                                                                    183,1-4       52%
```

입력된 값이 저장되어 있는지 ID를 통해 PrivateData에 접근하여 판별함. 만약 읽어온 값이 비어 있지 않다면 에러를 발생함. ClientIdentity 함수를 불러와서 ClientID 변수에 할당함. verifyClientOrgMatchesPeerOrg함수에서는 다른 조직에서 실행하는 것을 허용되지 않게 함. 다음으로 data와 private-data를 저장함.

```
180     // Save civil personal information to collection visible to owning organization
181     civilPrivateDetails := CivilPrivateDetails{
182         ID:         civilInput.ID,
183         Name:       civilInput.Name,
184         PhoneNumber:civilInput.PhoneNumber,
185         Address:    civilInput.Address,
186     }
187     civilPrivateDetailsAsBytes, err := json.Marshal(civilPrivateDetails) // marshal civil deta
    ils to JSON
188     if err != nil {
189         return fmt.Errorf("failed to marshal into JSON: %v", err)
190     }
191
192     // Get collection name for this organization.
193     orgCollection, err := getCollectionName(ctx)
194     if err != nil {
195         return fmt.Errorf("failed to infer private collection name for the org: %v", err)
196     }
197
198     // Put civil private value into client org specific private data collection
199     log.Printf("Put: collection %v, ID %v", orgCollection, civilInput.ID)
200     err = ctx.GetStub().PutPrivateData(orgCollection, civilInput.ID, civilPrivateDetailsAsByte
    s)
201     if err != nil {
202         return fmt.Errorf("failed to put civil Private details: %v", err)
203     }
204
205     // Save civil infection information to collection visible to civil, control organization
206     civilInfectionStatus := CivilInfectionStatus{
207         ID:         civilInput.ID,
208         Status:     civilInput.Status,
209     }
210     civilInfectionStatusAsBytes, err := json.Marshal(civilInfectionStatus) // marshal civil in
    fection status to JSON
211     if err != nil {
212         return fmt.Errorf("failed to marshal into JSON: %v", err)
213     }
214     log.Printf("CreateCivil InfectionStatus: collection: %v, ID %v, client %v", civilCollectio
    n, civilInput.ID, clientID)
215     err = ctx.GetStub().PutPrivateData(civilCollection, civilInput.ID, civilInfectionStatusAsB
    ytes)
216     if err != nil {
217         return fmt.Errorf("failed to put civil infection into private data collection: %v", er
    r)
218     }
219     return nil
220 }
                                                                    220,1         67%
```

Civil 조직에서 개인정보를 저장하는 CivilPrivateDetails를 만들어서 저장함. PutPrivateData함수에서 콜렉션 json파일 ID와 저장될 정보를 입력함. 개인의 전염병 현상을 '정상', '의심된', '감염된' 저장할 CivilInfectionStatus를 구조체를 생성해서 저장함.

```go
const civilCollectionName = "civilCollection"

type civilTransientInput struct {
    ID          string      `json:"civilID"`
    Name        string      `json:"civilName"`
    PhoneNumber string      `json:"civilPhoneNumber"`
    Address     string      `json:"civilAddress"`
    Status      string      `json:"civilStatus"`
}

func TestCreateCivilBadInput(t *testing.T) {
    transactionContext, chaincodeStub := prepMocksAsOrg1()
    civilTransferCC := chaincode.SmartContract{}

    // No transient map
    err := civilTransferCC.CreateCivil(transactionContext)
    require.EqualError(t, err, "civil not found in the transient map input")

    civilPropMap := map[string][]byte{
        "civil_properties": []byte("ill formatted property"),
    }
    chaincodeStub.GetTransientReturns(civilPropMap, nil)
    err = civilTransferCC.CreateCivil(transactionContext)
    require.Error(t, err, "Expected error: transient amp with incomplete civil data")
    require.Contains(t, err.Error(), "failed to unmarshal JSON")

    testCivil := &civilTransientInput{
        Status: "testfulcivil",
    }
    setReturnCivilPropsInTransientMap(t, chaincodeStub, testCivil)
    err = civilTransferCC.CreateCivil(transactionContext)
    require.EqualError(t, err, "civilID field must be non-empty string")

    testCivil = &civilTransientInput{
        ID:     "id1",
        Name:   "john",
    }
    setReturnCivilPropsInTransientMap(t, chaincodeStub, testCivil)
    err = civilTransferCC.CreateCivil(transactionContext)
    require.EqualError(t, err, "civilPhoneNumber field must be non-empty string")

    // case when civil exists, GetPrivateData returns a valid data from ledger
    testCivil = &civilTransientInput{
        ID:          "id1",
        Name:        "john",
        PhoneNumber:"010-1234-4567",
        Address:     "Wonchen-dong, Suwon-si",
                                                          97,1-4        46%
```

e2_list_enter.go의 CreateCivil 함수를 테스트하기 위해 e2_list_enter_test.go에 작성함. 잘못된 값이 입력되면 생각한 에러가 나오는지 확인함. 입력되는 정보가 없는 경우, 빈 값이 입력되는 경우, 주어진 값이 다 들어오지 않은 경우를 불러와서 출력되는지 확인한다. EqualError를 통해 출력되는 값을 비교함.

```
106 func TestCreateCivilSuccessful(t *testing.T) {
107     transactionContext, chaincodeStub := prepMocksAsOrg1()
108     civilTransferCC := chaincode.SmartContract{}
109     testCivil := &civilTransientInput{
110         ID:             "id1",
111         Name:           "John",
112         PhoneNumber:    "010-1234-3213",
113         Address:        "Wonchen-dong, Suwon-si",
114         Status:         "Normal",
115     }
116     setReturnCivilPropsInTransientMap(t, chaincodeStub, testCivil)
117     err := civilTransferCC.CreateCivil(transactionContext)
118     require.NoError(t, err)
119     //Validate PutPrivateData calls
120     calledCollection, calledId, _ := chaincodeStub.PutPrivateDataArgsForCall(0)
121     require.Equal(t, civilCollectionName, calledCollection)
122     require.Equal(t, "id1", calledId)
123
124     expectedPrivateDetails := &chaincode.CivilPrivateDetails{
125         ID:             "id1",
126         Name:           "John",
127         PhoneNumber:    "010-1234-3213",
128         Address:        "Wonchen-dong, Suwon-si",
129     }
130     civilBytes, err := json.Marshal(expectedPrivateDetails)
131     calledCollection, calledId, calledCivilBytes := chaincodeStub.PutPrivateDataArgsForCall(1)
132     require.Equal(t, myOrg1PrivCollection, calledCollection)
133     require.Equal(t, "id1", calledId)
134     require.Equal(t, civilBytes, calledCivilBytes)
135 }
136
137 func TestVerifyIsPermitted(t *testing.T) {
138     transactionContext, chaincodeStub := prepMockAsOrg1()
139     civilTransferCC := chaincode.SmartContract{}
140 }
141
142 func setReturnCivilPropsInTransientMap(t *testing.T, chaincodeStub *mocks.ChaincodeStub, testC
    ivil *civilTransientInput) []byte {
143     civilBytes := []byte{}
144     if testCivil != nil {
145         var err error
146         civilBytes, err = json.Marshal(testCivil)
147         require.NoError(t, err)
148     }
149     civilPropMap := map[string][]byte{
150         "civil_properties": civilBytes,
151     }
                                                                          151,1-4       97%
```

CreateCivil에서 제대로 값이 들어왔을 때 생성되는 것을 확인함. 제대로 저장되어 들어간 값이 정확하게 리턴 값을 비교함. 문제가 없으면 출력을 내고 종료됨. 이후 작성할 TestVerfifyIsPermitted가 지정됨. setReturnCivilPropsInTransientmap을 통해 캡슐화해서 반복적으로 불릴 함수를 구조화함.

```
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go/chaincode$ go test
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 id1 does not exist in collection assetCollection
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 ReadAssetPrivateDetails: collection Org1TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 AssetPrivateDetails for id1 does not exist in collection Org1TestmspPrivateCol
lection
2021/05/16 14:04:06 ReadAssetPrivateDetails: collection Org2TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 ReadAssetPrivateDetails: collection Org1TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 ReadTransferAgreement: collection assetCollection, ID id1
2021/05/16 14:04:06 TransferAgreement for id1 does not exist
2021/05/16 14:04:06 ReadTransferAgreement: collection assetCollection, ID id1
Asset already exists: id1
2021/05/16 14:04:06 CreateAsset Put: collection assetCollection, ID id1
2021/05/16 14:04:06 Put: collection Org1TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 id1 does not exist in collection assetCollection
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 AgreeToTransfer Put: collection Org1TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 AgreeToTransfer Put: collection assetCollection, ID id1, Key transferAgreement
id1
2021/05/16 14:04:06 TransferAsset: verify asset exists ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 id1 does not exist in collection assetCollection
2021/05/16 14:04:06 TransferAsset: verify asset exists ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 ReadTransferAgreement: collection assetCollection, ID id1
2021/05/16 14:04:06 TransferAsset Put: collection assetCollection, ID id1
2021/05/16 14:04:06 TransferAsset: verify asset exists ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 TransferAsset: verify asset exists ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
2021/05/16 14:04:06 ReadTransferAgreement: collection assetCollection, ID id1
2021/05/16 14:04:06 TransferAsset: verify asset exists ID id1
2021/05/16 14:04:06 ReadAsset: collection assetCollection, ID id1
Civil already exists: id1
2021/05/16 14:04:06 CreateCivil Put: collection: civilCollection, ID id1, client ◆#◆◆U,z◆◆
2021/05/16 14:04:06 Put: collection Org1TestmspPrivateCollection, ID id1
2021/05/16 14:04:06 CreateCivil InfectionStatus: collection: civilCollection, ID id1, client ◆#◆◆U
,z◆◆
PASS
ok      github.com/hyperledger/fabric-samples/asset-transfer-private-data/chaincode-go/chaincode 0
.040s
h2kim@h2kim-VBox:~/fabric-samples/e2-list-private-data/chaincode-go/chaincode$
```

$go test를 입력하여 아직 지우지 않은 asset_quries_test.go와 asset_transfer_transfer_test.go를 테스트한 이후 앞에 작성한 CreateCivil 테스트를 한 로그가 출력됨.

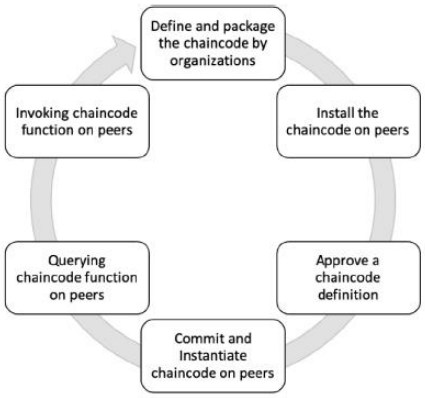


Fig. 4. Fundamental operations related to the Chaincode.

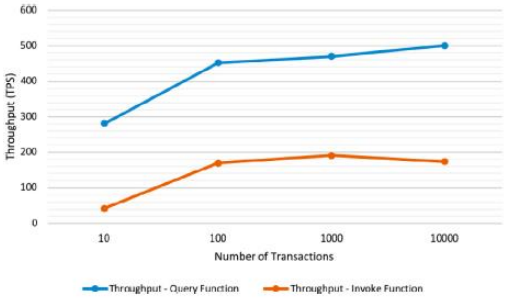


Fig. 5. Throughput for Query and Invoke functions.

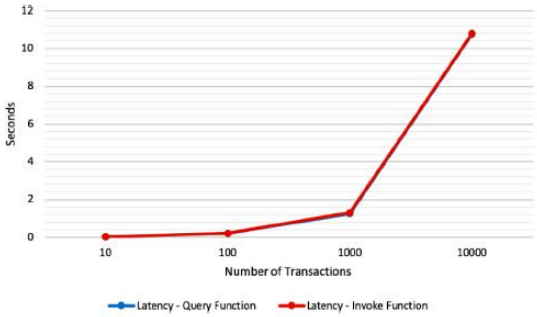


Fig. 6. Latency for Query and Invoke functions.

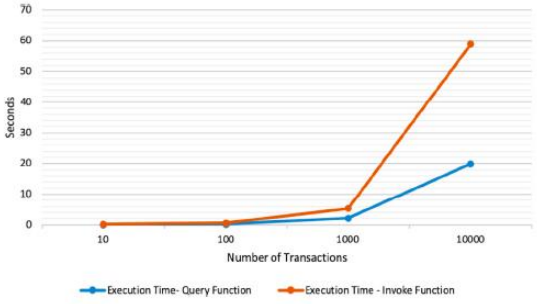


Fig. 7. Execution time for Query and Invoke functions.

참고자료[1]의 에너지 거래소에서 하이퍼레저 패브릭을 이용하는 논문을 참고했습니다. Fig 4는 하이퍼레저 패브릭이 작동하는 프로세스를 도표화했습니다.

Table 6
Description of notable input parameters and operations for the proposed algorithms.

Algorithms	Chaincode methods	Transactions	Frequency	Endorsement policy	Database
Application account opening	Init, createAsset	Update Query Transfer	Six Null Null	AND ('O1.m', 'O2.m', 'O3.m') or OutOf (2, 'O1.m', 'O2.m', 'O3.m')	LevelDB or CouchDB
Receiver-initiated energy trading	Init, getAsset, createAsset, getAssetsFromBatch	Update Query Transfer	Five Two One		
Sender-initiated energy trading	Init, getAsset, createAsset, getAssetsFromBatch	Update Query Transfer	Four One One		

Table 7
Description of configurations utilized in our experimentation.

Performance attributes		Batch timeout (in ms)	Maximum message count	Number of transactions	Transaction arrival rate	Number of clients
Configuration	A	300	100	200	120	10
	B	600	200	400	240	15
	C	900	300	600	360	20
	D	1200	400	800	480	25
	E	1500	500	1000	600	30
	F	1800	600	1200	720	35
	G	2100	700	1400	840	40

알고리즘에 관한 설명과 작동하는 곳의 설정 데이터베이스를 설명한 후 적용되는 설정을 표로 넣은 것을 볼 수 있습니다. 처리량 지연율 실행 시간을 구하는 것을 볼 수 있습니다. 트랜잭션의 크기를 가지고 비교하는 것을 볼 수 있었습니다.

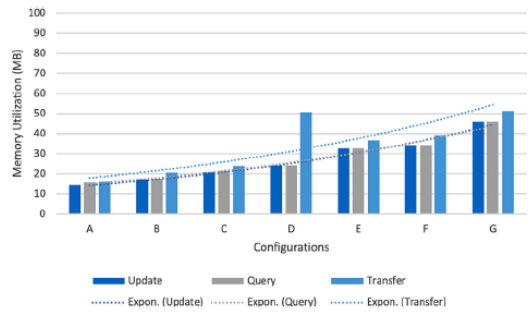


Fig. 8. Memory utilization by Orderer in Hyperledger Fabric v1.4.0 for various configurations (described in Table 7).

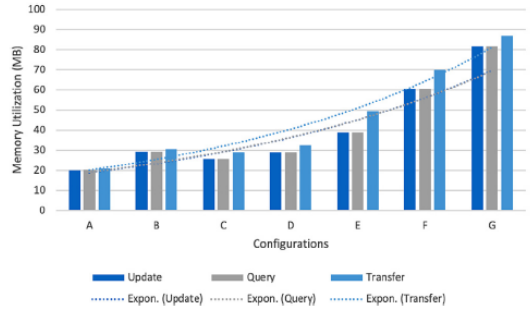


Fig. 9. Memory utilization by Orderer in Hyperledger Fabric v1.4.1 for various configurations (described in Table 7).

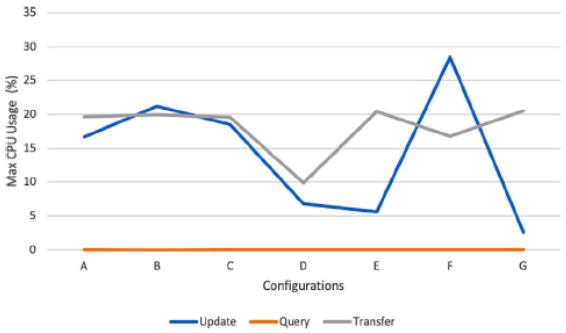


Fig. 10. Maximum CPU utilization of Orderer in Hyperledger Fabric v1.4.0.

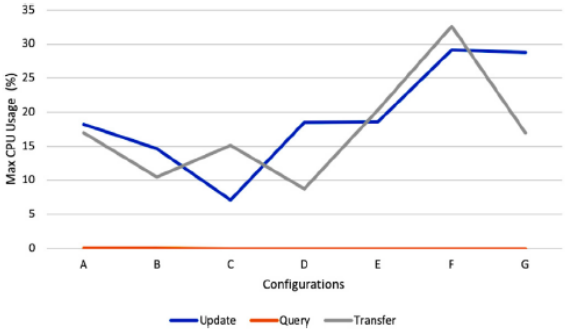
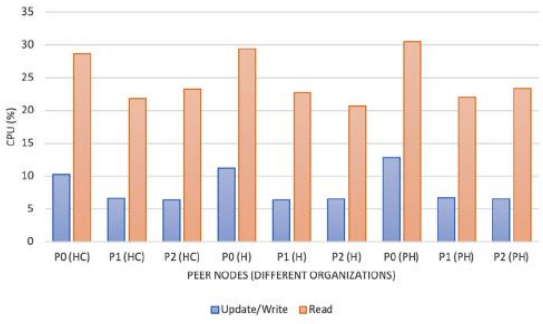
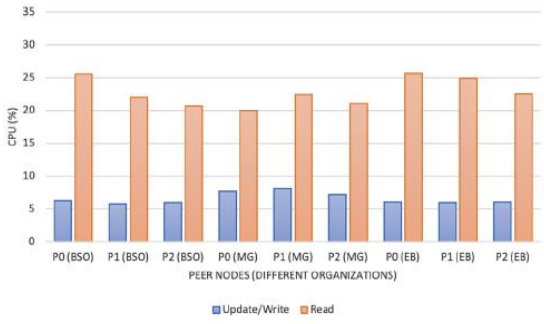


Fig. 11. Maximum CPU utilization of Orderer in Hyperledger Fabric v1.4.1.

환경설정을 한 것에 따라서 메모리와 CPU 사용량을 측정하는 것을 볼 수 있었습니다.



(a)



(b)

피어에 따라서 다른 조직에 있을 때에 업데이트/쓰기 또는 읽기 할 때 CPU 사용율을 측정한 것을 볼 수 있습니다.

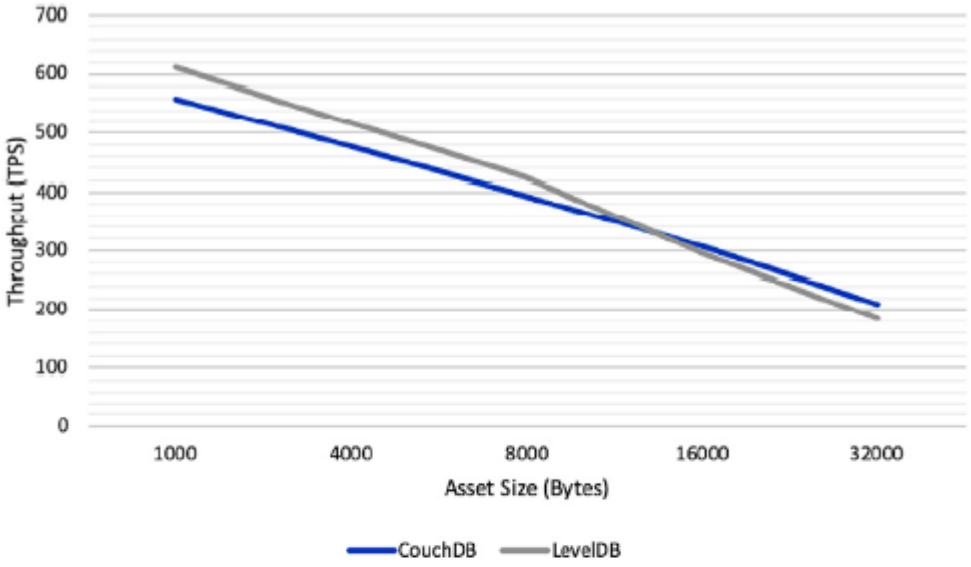


Fig. 13. Throughput of Query transaction within CouchDB and LevelDB-based networks.

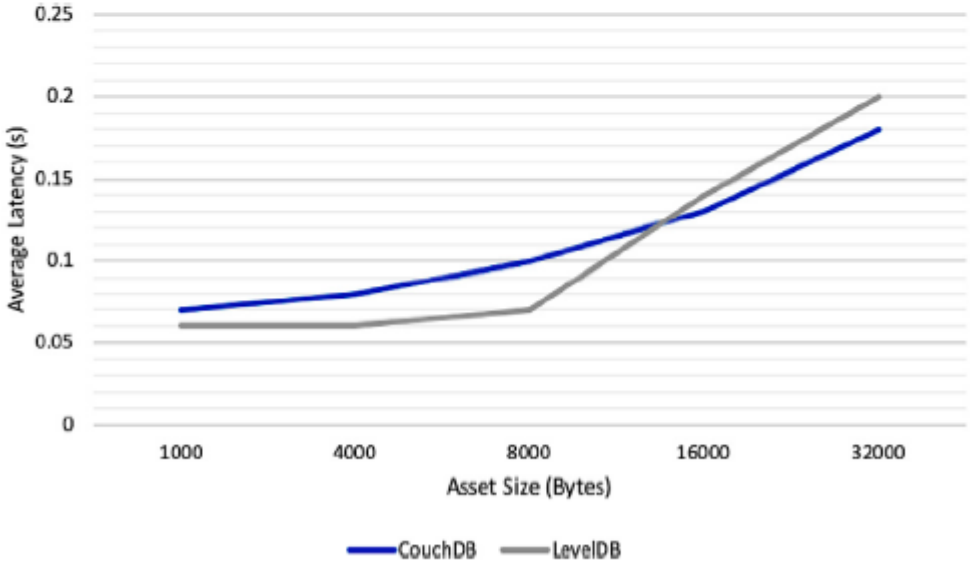


Fig. 14. Average latency of Query transaction within CouchDB and LevelDB-based networks.

Fig. 14에서는 처리률 지연율을 CouchDB와 LevelDB에 따라서 구한 값을 그래프로 표현한 것을 볼 수 있었습니다.

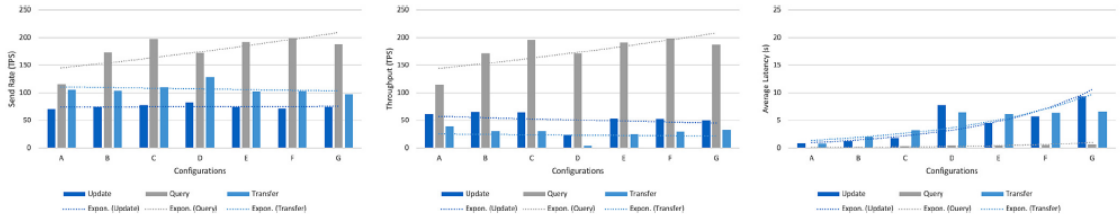


Fig. 15. Send rate, throughput, and average latency investigated for various configurations (described in Table 7) in HF v1.4.0.

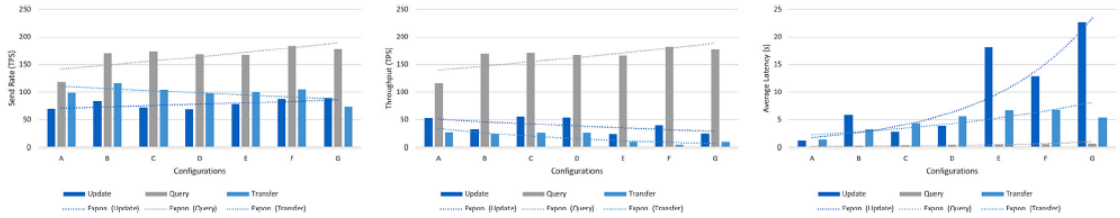


Fig. 16. Send rate, throughput, and average latency investigated for various configurations (described in Table 7) in HF v1.4.1.

함수를 실행하는 것에 따라서 전송, 처리, 평균지연시간을 측정하는 것을 확인할 수 있었습니다.

참고자료[2] 하이퍼레저 기반 기술을 이용한 환자중심의 의료보건시스템을 제안하고 있습니다. 코로나-19 유행병에 따라서 새로운 시스템이 필요하다고 이야기하고 있습니다.

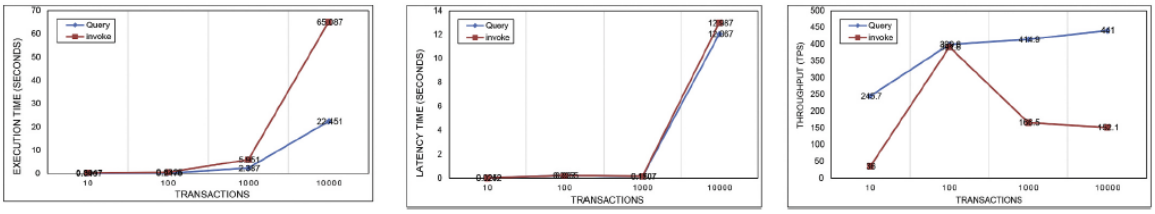


Fig. 4. Execution cost, latency cost and throughput of query and invoke phase for varying (up to 10,000) transactions.

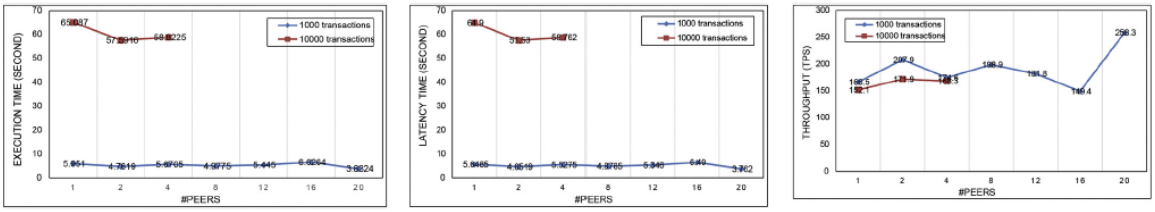


Fig. 5. Execution cost, latency cost and throughput of 1000 and 10,000 transactions for different (upto 20) peers.

Fig 4 에서는 트랜잭션에 따라서 실행시간, 지연시간, 전송율을 측정하는 것을 볼 수 있습니다. Fig 4.에서는 피어의 수 따라서 트랜잭션을 다리게 하면서 실행시간, 지연시간, 전송율을 측정한 것을 확인했습니다.

성능평가는 2개의 조직 4개의 조직으로 지연율을 측정했습니다.

참고자료[3]에서는 안전한 작업공유를 할 때 블록체인을 사용하는 것을 제안하고 있습니다. 다중공유를 통할 때에 에지(Edge)컴퓨팅을 사용하는 것을 가정하고 있습니다.

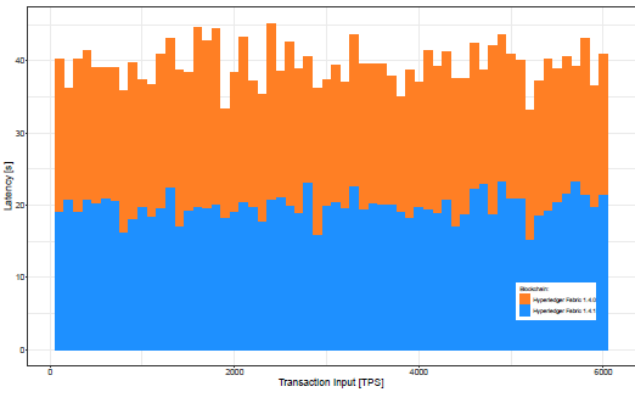


Fig. 4: Variation of transaction latency versus input transaction rate for Hyperledger Fabric versions 1.4.0 and 1.4.1.

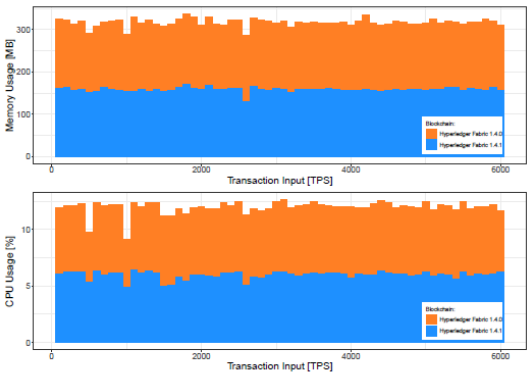


Fig. 5: Variations of CPU usage & memory usage with input transaction rate for Hyperledger Fabric versions 1.4.0 and 1.4.1.

Hyperledger 1.4.0 1.4.1 버전에 따라서 지연율, CPU 사용량, 메모리 사용량을 측정한 것을 볼 수 있습니다.

3. 차주 계획

- 체인코드 함수 작성 테스트코드 작성
- 논문 검색 후 타 논문 연구 검색지속

4. 참고 자료

[1] Ankur Lohachab, Saurabh Garg, Byeong Ho Kang, Muhammad Bilal Amin "Performance evaluation of Hyperledger Fabric-enabled framework for pervasive peer-to-peer energy trading in smart Cyber–Physical Systems" Future Generation Computer Systems Volume 118, pp 392-416

[2] Mahender Kumar, Satish Chand "MedHypChain: A patient-centered interoperability hyperledger-based medical healthcare system: Regulation in COVID-19 pandemic" Journal of Network and Computer Applications Volume: 179 pp. 1-14

[3] Rivera Angelo Vera, Refaey Ahmed, Hossain Ekram "A Blockchain Framework for Secure Task Sharing in Multi-access Edge Computing" access url: http://arxiv.org/abs/2006.14166 pp 1-7