IBM

Open by design™

IBM开源技术微讲堂

区块链和HyperLedger系列

第五讲

Hyperledger中的共享账本

http://ibm.biz/opentech-ma

“区块链和HyperLedger” 系列公开课

- 每周四晚8点档
  - 区块链商用之道
  - HyperLedger review
  - HyperLedger架构解读
  - **HyperLedger 中的共享账本**
  - HyperLedger中的共识管理
  - HyperLedger中的隐私与安全
  - HyperLedger应用案例赏析

讲师介绍—贾锡学

- IBM中国系统实验室
- Hyperledger开源社区爱好者
- Bluemix Blockchain service技术支持
- 参与国内金融保险行业Blockchain技术支持及PoC

议程

- What is shared ledger
- Ledger for Fabric v1.0
- Ledger privacy with multiple channels
- Ledger and Chaincode

What is shared ledger

Blockchain is **A shared ledger technology allowing any participant in the business network to see THE system of record (ledger)**

Ledger provides a verifiable history of all successful state changes. It is THE system of record for a business. Business will have multiple ledgers for multiple business networks in which they participate. The ledger is SHARED, REPLICATED and PERMISSIONED.

Transaction – an asset transfer onto or off the ledger

John gives a car to Anthony (simple)

Contract – conditions for transaction to occur

If Anthony pays John money, then car passes from John to Anthony (simple)

If car won't start, funds do not pass to John (as decided by third party arbitrator) (more complex)

Ledger for Fabric v1.0

Block ledger

- File system based, only new Blocks appending
- Blocks are stored on all committers and optional subset of ordering service nodes

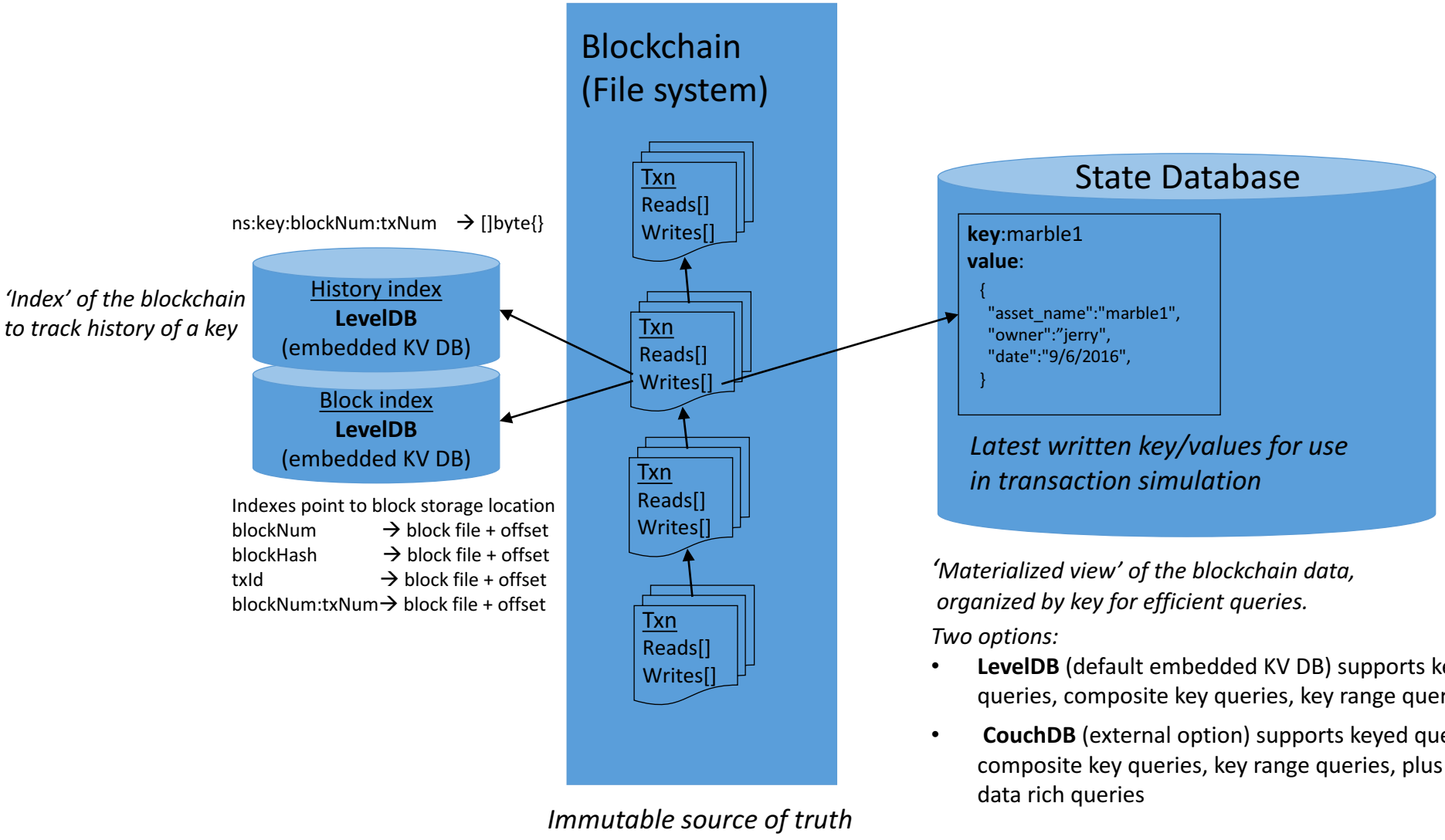
State ledger

- World/Ledger state holds current value of smart contract data
  e.g. vehicleOwner=Daisy
- KVS hidden from developer by chaincode APIs
  e.g. GetState(), PutState(), GetStateByRange(), etc…
- Stored on all committers

History ledger

- Holding historic sequence of all chain code transactions
  e.g. updateOwner(from=John, to=Anthony); updateOwner (from=Anthony, to=Daisy);etc
- Index stored in KVS and hidden from developer by chaincode APIs
  e.g. GetHistoryForKey()
- Stored on all committers

Ledger for Fabric v1.0



Immutable source of truth

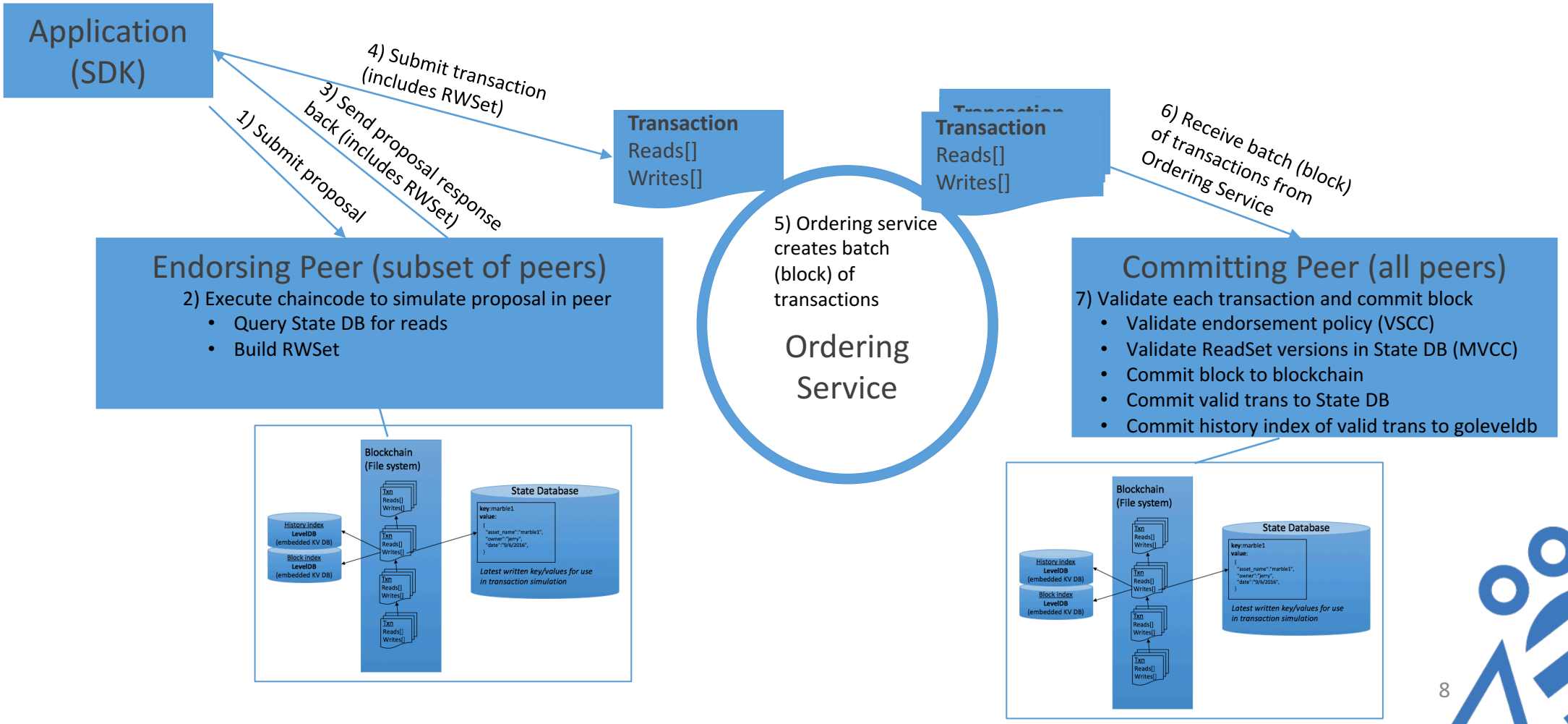
'Materialized view' of the blockchain data, organized by key for efficient queries.

Two options:

- **LevelDB** (default embedded KV DB) supports keyed queries, composite key queries, key range queries
- **CouchDB** (external option) supports keyed queries, composite key queries, key range queries, plus full data rich queries ***Beta in v1***

Ledger for Fabric v1.0

Transaction lifecycle and interaction with ledgers



Ledger for Fabric v1.0

Logical structure of a ReadWriteSet

```
Block{
   Transactions [
      {
         "Id" : txUUID2
         "Invoke" : “Method(arg1, arg2,..,argN)"
         “TxRWSet" : [
                        { ”Chaincode” : “ccId”
                                 “Reads”:[{"key" : “key1", "version” : “v1” }]
                                 “Writes”:[{"key" : “key1", ”value" : bytes1}]
                      } // end chaincode RWSet
                   ] // end TxRWSet
       },  // end transaction with "Id" txUUID2

      { // another transaction },
   ] // end Transactions
}// end Block
```

Endorsing Peer (Simulation):

- Simulates transaction and generates ReadWriteSet

Committing Peer (Validation/Commit):

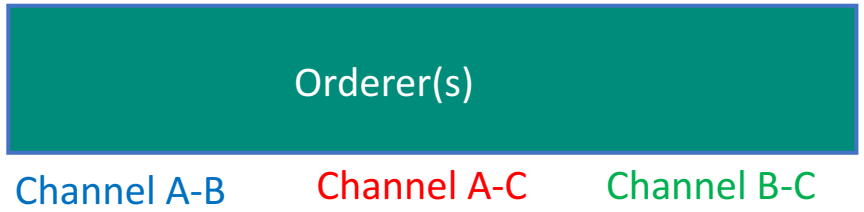
- Read set is utilized by MVCC validation check to ensure values read during simulation have not changed (ensures serializable isolation).
- Block is added to chain and each valid tran’s Write Set is applied to state database; history index of valid trans committed

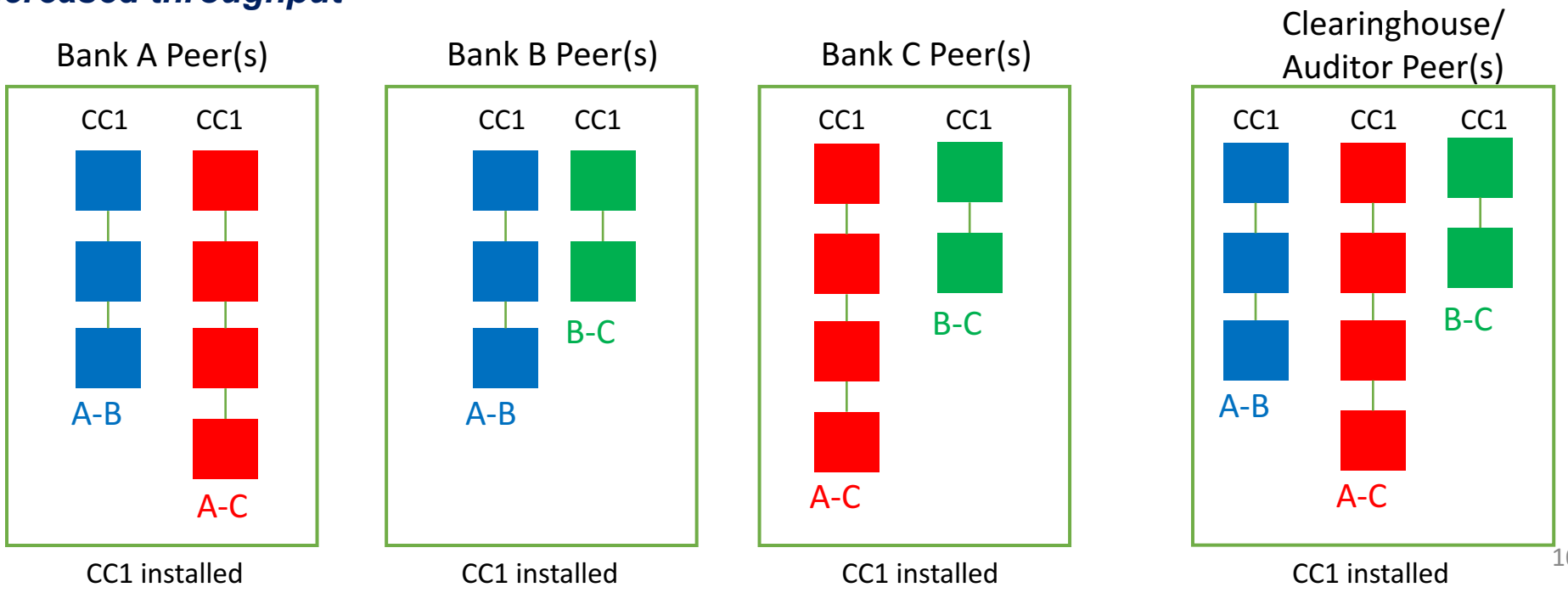
Ledger privacy with multiple channels

Chaincode1 installed on all 4 peers.
Chaincode1 instantiated on all 3 channels*
Different chaincodes could be instantiated on different channels.
**Multiple chaincodes can be instantiated on each channel.*
One distributed ledger per channel.

Bank A cannot see transactions between B and C.
Blocks from different channels can be processed in parallel.

➔ ***Privacy + increased throughput***





Ledger and Chaincode

```
type Chaincode interface {
    Init(stub ChaincodeStubInterface) pb.Response
    Invoke(stub ChaincodeStubInterface) pb.Response
}

type ChaincodeStubInterface interface {
    ......
    GetState(key string) ([]byte, error)
    PutState(key string, value []byte) error
    DelState(key string) error
    GetStateByRange(startKey, endKey string) (StateQueryIteratorInterface, error)
    GetStateByPartialCompositeKey(objectType string, keys []string) (StateQueryIteratorInterface, error)
    GetQueryResult(query string) (StateQueryIteratorInterface, error)
    GetHistoryForKey(key string) (StateQueryIteratorInterface, error)
    ......
}
```

Ledger and Chaincode

Single key operations

GetState()/PutState()/DelState() - Read, Write and Delete a single key/value.

Key range queries

Can be used in chaincode transaction logic (e.g. identify keys to update).

Fabric guarantees result set is stable between endorsement time and commit time, ensuring the integrity of the transaction.

GetStateByRange()

- Read keys between a startKey and endKey.

GetStateByPartialCompositeKey()

- Read keys that start with a common prefix. For example, for a chaincode key that is composed of K1-K2-K3 (composite key), ability to query on K1 or K1-K2 (performs range query under the covers). Replacement for v0.6 GetRows() table api.

Non-Key queries on data content beta in v1

Available when using a state database that supports content query (e.g. CouchDB)

Read-only queries against current state, not appropriate for use in chaincode transaction logic, unless application can guarantee result set is stable between endorsement time and commit time.

GetQueryResult()

- Pass a query string in the syntax of the state database

Ledger and Chaincode

Query System Chaincode(QSCC)

- New system chaincode deployed by default in v1 to query blockchain
- Client can invoke against any peer, using same endorser request/response model that is used for application chaincode calls
- QSCC includes the following APIs (chaincode functions):
  - GetChainInfo
  - GetBlockByNumber
  - GetBlockByHash
  - GetTransactionByID – returns the processed transaction as well as valid/invalid indicator

IBM

Open by design™

IBM开源技术微讲堂

区块链和HyperLedger系列

第五讲完

http://ibm.biz/opentech-ma