- **Design Process for Applications on Blockchain**
- **Cost**

COMP6452 Lecture 5.1: Design Process for Applications on Blockchain

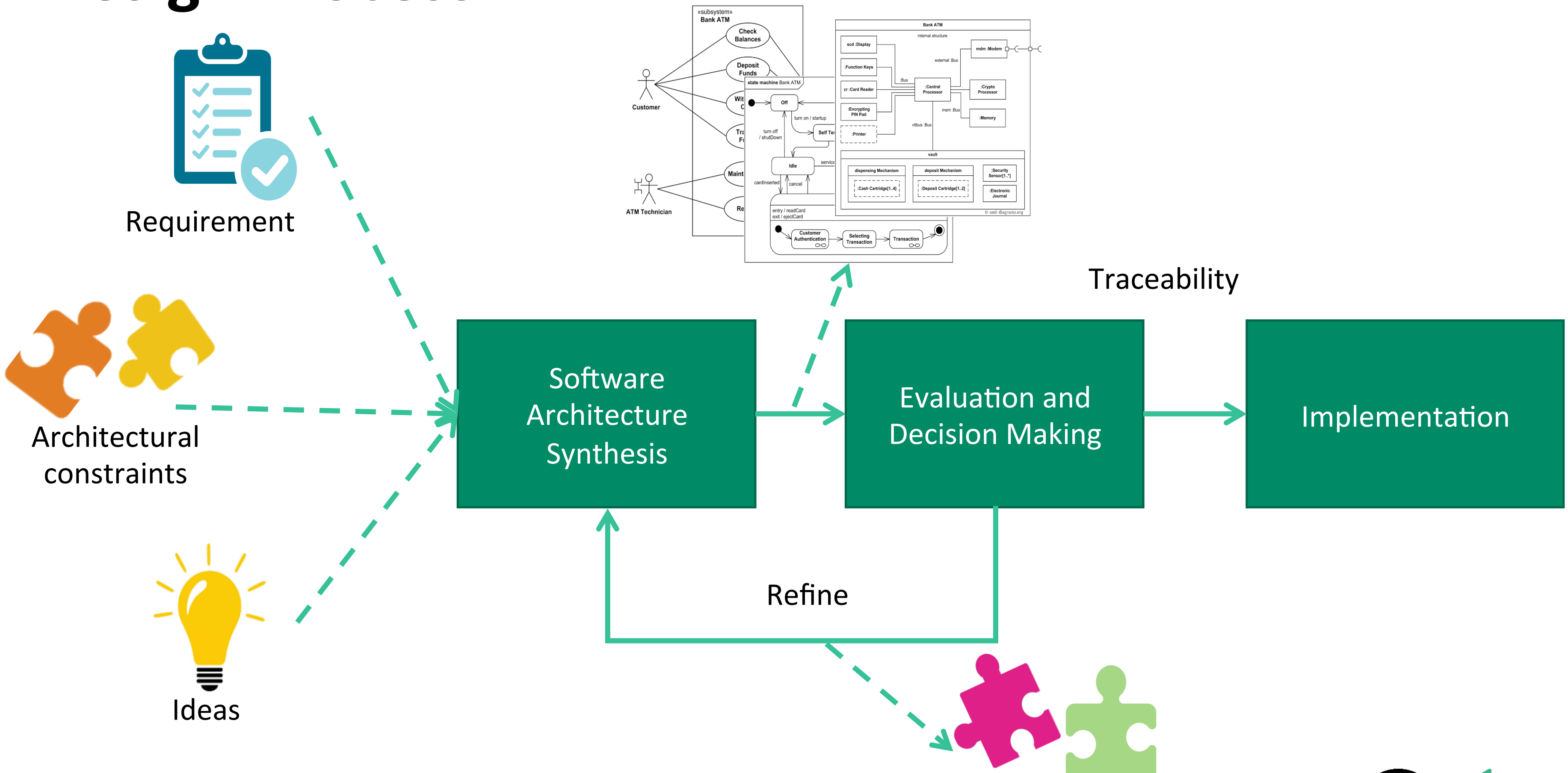
Xiwei (Sherry) Xu (xiwei.xu@data61.csiro.au)

18th of March, 2019

Outline

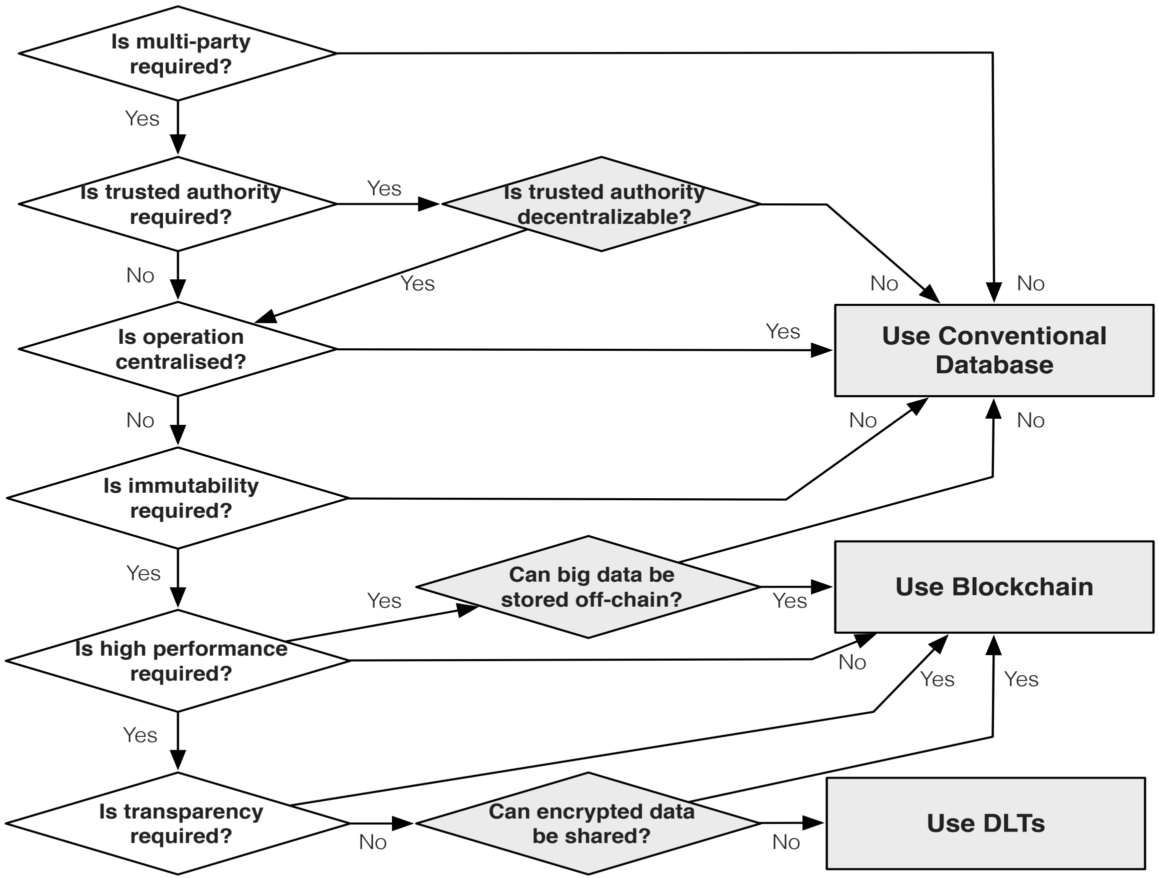
- Evaluation of Suitability
- Example Use Cases for Suitability Evaluation
- Design Process for Blockchain-based Systems
- Subsidiary Design Choices
    - Private blockchain vs. public blockchain
    - What consensus protocol
    - What the block frequency
- What's on-chain and What's off-chain

Design Process

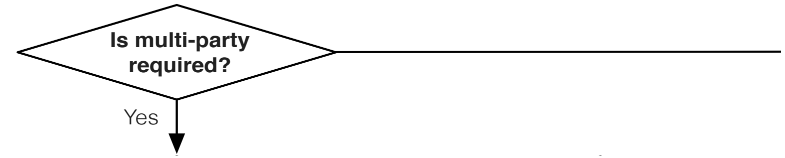


Evaluation of Suitability

Evaluation Framework



Is Multi-party Required? 1/2



- Blockchain is **NOT SUITABLE** for systems that only serve individual isolated users
  - Conventional database is simpler and more efficient

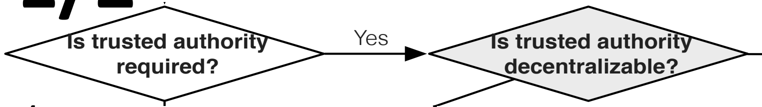
- Different legally distinct parties
  - Supply chain
    - Complex, dynamic, multi-party arrangements with regulatory and logistical constraints spanning jurisdictional boundaries
      - Manufacturers, shipping companies, transport infrastructure organizations, financial service firms, or regulators
    - Information exchange can be as important and difficult as the physical exchange of goods
  - Inter-bank payments and reconciliation
    - Two different banks
    - Account holders are organizations or individuals

Is Multi-party Required? 2/2

- Large enterprise or government
  - Different functional or geographic divisions or departments
  - Informational or administrative “silos” as multiple parties

- Blockchain can be **SUITABLE** for supporting multi-party systems
  - Physically distributed
  - Logically centralized
  - Infrastructure providing a single view of truth across those parties

Is Trusted Authority Required? 1/2



- An entity that is relied upon to perform a function (operating a system)
  - Blockchain is **NOT SUITABLE** if a single party can or must be relied upon as a trusted authority
  - Trusted authority implements a traditional centralized solution with conventional technologies
    - Banks
    - Government departments

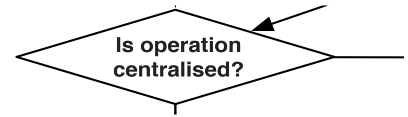
- Scope of the system is important in deciding the question
  - Bank is a trusted authority for payment transfer among bank accounts
  - Central bank is a trusted authority for inter-bank payments
    - The two banks collective rely upon central bank

- Trusted authority is a single point of failure
  - Technical single points of failure can be mitigated by using redundancy
  - Single points of organizational or business failure remain present
    - Business failures, service interruptions, data loss or fraud

Is Trusted Authority Required? 2/2

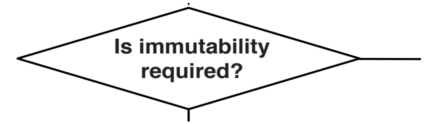
- Trusted authority is a monopoly or oligopoly service provider
  - "rent-seeking" behavior
    - Increase one's share of existing wealth without creating new wealth
- A natural trusted authority might be difficult for everyone to accept reliance on that party
  - Organization could define a central agency to provide services for coordinated operations across the whole organization
  - Centralization of services can be perceived as a loss of control or power
- Blockchain is **SUITABLE** for system with no single party that is acceptable for operating the system
  - Operated jointly by a collective of nodes
  - Not remove trust
    - Users are exposed to risk in use of blockchain technology
    - What is trusted is the software, the incentive mechanism, and "oracles"
  - Distributed Trust
    - Remove the need to trust a single third-party to maintain a ledger

Is operation centralized?



- System with multiple parties, but no party is suitable as a trusted authority for administering the system
  - Group of parties form a joint venture to operate a conventional centralized system
  - Credit card associations, like Visa and MasterCard
    - Joint venture between banks
- Centralized operation of the system lead to the administering party becoming a trusted authority
  - Unacceptable to the parties within the system
  - Forming a new entity like a joint venture is too costly
  - Centralized administration may cause single point of business failure
- Blockchain-based systems do not need a single system operator
  - Better system reliability and availability

Is Immutability Required? 1/3



- *Data immutability* means data can not be changed or altered after its creation
  - Immutability supports non-repudiation
    - Assurance that a party cannot deny the authenticity of their signature
- Conventional technologies naturally support mutable data
- Blockchain naturally supports data immutability in the ledger
  - Linking of blocks in a chain of cryptographic hashes supports immutability
  - Data continually replicated across many locations and organizations
    - Attempts to change it in one location will be interpreted as an attack on integrity
  - Strong evidence that the transactions were performed by someone with control over their cryptographic keys
- Transaction history is immutable
  - Transaction changes the latest view of the current state

Is Immutability Required? 2/3

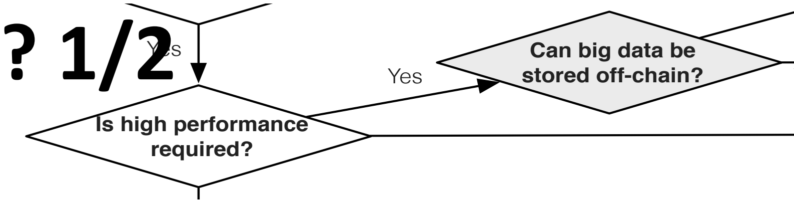
- Impossible to change the transaction history in most blockchains
  - Cause problems if blockchain contains illegal content
  - Court orders content to be removed from the blockchain
- Immutability makes it less adaptable for the following issues
  - Disputed transactions
  - Incorrect addresses
  - Exposure or loss of private keys
  - Data-entry errors
  - Unexpected changes to assets tokenized on blockchain
- Other cheaper mechanisms available to prove the originality of data
  - Hashing technology
  - Cryptographically signed data

Is Immutability Required? 3/3

- Immutability of PoW-based blockchain is a long-run probabilistic durability
  - Conventional database supports ACID (Atomicity, Consistency, Isolation and Durability)
  - A transaction initially thought by a participant to be committed may later turn out to have been on a shorter chain
  - A transaction is in practice be immutable if it has been committed to a blockchain for a sufficiently long time (number of blocks)
    - X-confirmation *(Lecture 7)*

- Blockchain using other consensus mechanism can offer stronger and more conventional immutability
  - E.g. PBFT(Practical Byzantine Fault Tolerance)
  - Small number of known nodes participating in the operation

Is High Performance Required? 1/2



- Blockchain is ***NOT SUITABLE*** if the system need to support high performance
  - Extremely short response time (Latency)
  - Process very large amounts of data (Throughput)
- Bitcoin and Ethereum cannot currently match the maximum throughput of conventional transaction processing system
  - Visa payments network
- New mechanisms to improve performance (*Lecture 3*)
  - Sharding
  - State channels
  - Reduced inter-block time
- Consortium and private blockchains with careful design and performance tuning have much better performance

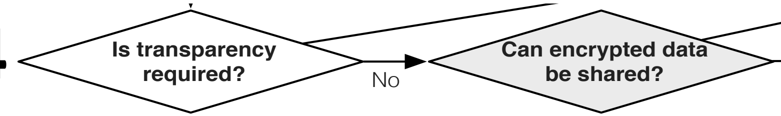
Is High Performance Required? 2/2

- Read latency can be much faster than conventional technologies
  - Response time for accessing historical data from a blockchain client
  - Clients keep a full local copy of the database
  - No network delays

- Write latency is probabilistic with several sources of uncertainty
  - Network delay of transaction propagation
  - Consensus process delay
  - Confirmation blocks on PoW-based blockchain increases write latency

- Blockchain is **NOT SUITABLE** for storing Big Data
  - Large volumes or high velocity
  - Massive redundancy

Is Transparency Required? 1/4



- *Data transparency* means data is available and accessible to by other parties
  - Facebook public newsfeed posts
  - Twitter public tweets
  - Facebook/Twitter support confidentiality
    - Choose what content publish to the public or to specific audience

- Blockchain provides a neutral platform where all participants can see and audit the published data
  - Validation of cryptocurrency transfers
    - From addresses with enough cryptocurrency
    - Signed with an authorized private key
  - Validation of smart contract execution is correctly recorded on the blockchain

Is Transparency Required? 2/4

- Blockchain **MAY BE SUITABLE** if data transparency is required or acceptable
- Confidentiality is harder to establish in blockchain-based systems
- Information is visible to all participants

- Amount of interactions between parties is confidentiality concern
- Very often customer relationships, pricing, or aggregated transaction volume are commercially-sensitive information
- Use pseudonymity
  - Contents of a transaction are publicly visible
- Create a new address for each transaction
  - Flow of assets may be used to infer relationships between addresses
- Reidentification
  - Reuse of addresses and their connection via transfers of cryptocurrency

Is Transparency Required? 3/4

- Public blockchain **MAY BE SUITABLE**
  - Public advertising or fully open government registries in highly regulated industries
    - Banks advertise on television
    - Television is not a highly-regulated banking transaction system
    - Data integrity and publicity is required rather than privacy or confidentiality
  - Secure software package management
  - IoT device configuration updates
- Blockchain can be used to share encrypted data
  - Asymmetrical with a party's public key
  - Symmetrical with a shared secret key
    - Requires a secure means of exchanging the secret key
  - Increase confidentiality, but reduce performance
  - Encrypted data makes it difficult to use smart contracts with the data
    - Embedding keys within a smart contract would reveal the keys

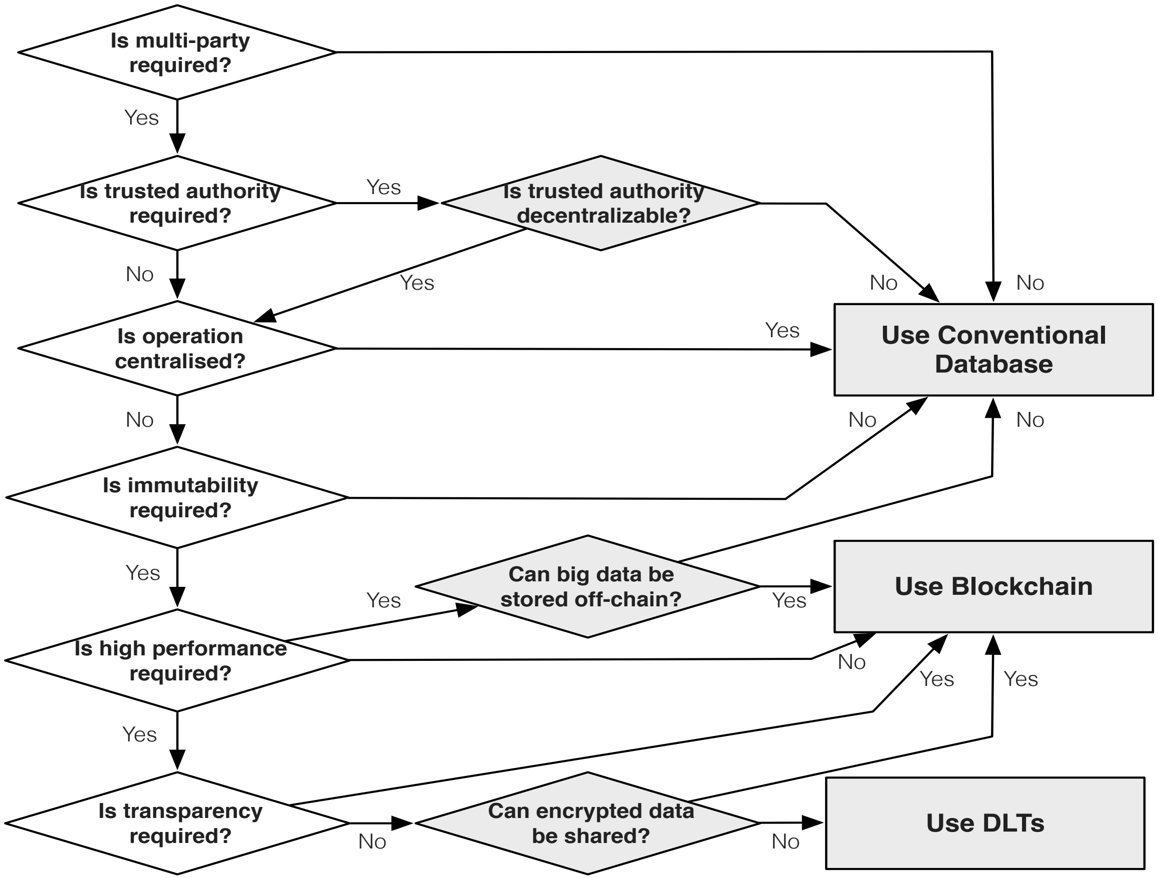
Is Transparency Required? 4/4

- Sometimes encryption is not acceptable
  - Concerns about successful encryption key management
  - Future technological developments in decryption (e.g. quantum computing)
  - Reveal information as meta-data

- Consortium and private blockchains provide read access controls
  - Data is not commercial confidentiality between competitors
  - Trade-of is between the benefits of sharing data within the group of collaborators and retaining confidentiality towards competitors where needed

- More controlled data sharing can be enabled by distributed ledger technology
  - Corda or Hyperledger Fabric
    - Small ledgers shared between parties of interest to store each transaction

Evaluation Framework



Example Use Cases for Suitability Evaluation

	Supply chain	Electronic health records	Identity	Stock market
Multi-party	Required	Required	Required	Required
Trusted authority	Not required	Decentralized	Not required	Not required
Centralized operation	Not required	Not required	Not required	Not required
Data immutability & Non-repudiation	Required	Required	Required	Required
High performance	Not required	Not required	Not required	Required
Data transparency & Confidentiality	Transparent (but not fully public)	Confidential	Transparent	Confidential
Sample Result	DLT	Conventional System	Blockchain	Conventional System

Use Case 1: Supply Chain 1/2

- A collection of processes involved in creating and distributing goods, from raw materials to completed products, through to consumers
- Highly complex **multi-party system**
  - Farmers, factories, transport providers, and retailers
- **Operations** are distributed and loosely coupled
- **Data transparency** is desired by participants
  - Support logistics planning, and to identify and respond to problems
  - Controlled confidentiality is required
    - Small ledgers between parties in interest
    - Conventional information exchange + hashed information on blockchain
- Transaction history and **data immutability** are desired to enable traceability back to the origin of goods
  - Control fraud and substitution

Use Case 1: Supply Chain 2/2

	Supply chain
Multi-party	Required
Trusted authority	Not required
Centralized operation	Not required
Data immutability & Non-repudiation	Required
High performance	Not required
Data transparency & Confidentiality	Transparent (but not fully public)
Sample Result	DLT

- The time taken in a supply chain is dominated by physical transportation and storage, which moderates demand for **performance**
  - Reasonably short latency is required at key points of hand-over of goods
- Dynamic structure of business relationship and **operation** can be accommodated by blockchain network
- Logically-centralized view of information supports demands for transparency

Use Case 2: Electronic Health Records 1/4

- Collections of patient medical records
  - Blood type, vital signs, past medical records, medication, and radiology report
  - Maintained by specific healthcare providers in siloed systems
- **Multiple parties** from different medical jurisdictions are involved
  - Patients, professionals and organizations
- Healthcare service providers are **decentralized trusted authorities**
  - Each has access to patient data and authority to make changes
- **Operation is distributed** across healthcare service providers
- **Data transparency** is the main issue
  - Patient privacy
  - Shared with patient consent
  - Exceptions: emergency situations; access to anonymised data for approved medical research
- Health records cannot be inappropriately created or updated

Use Case 2: Electronic Health Records 2/4

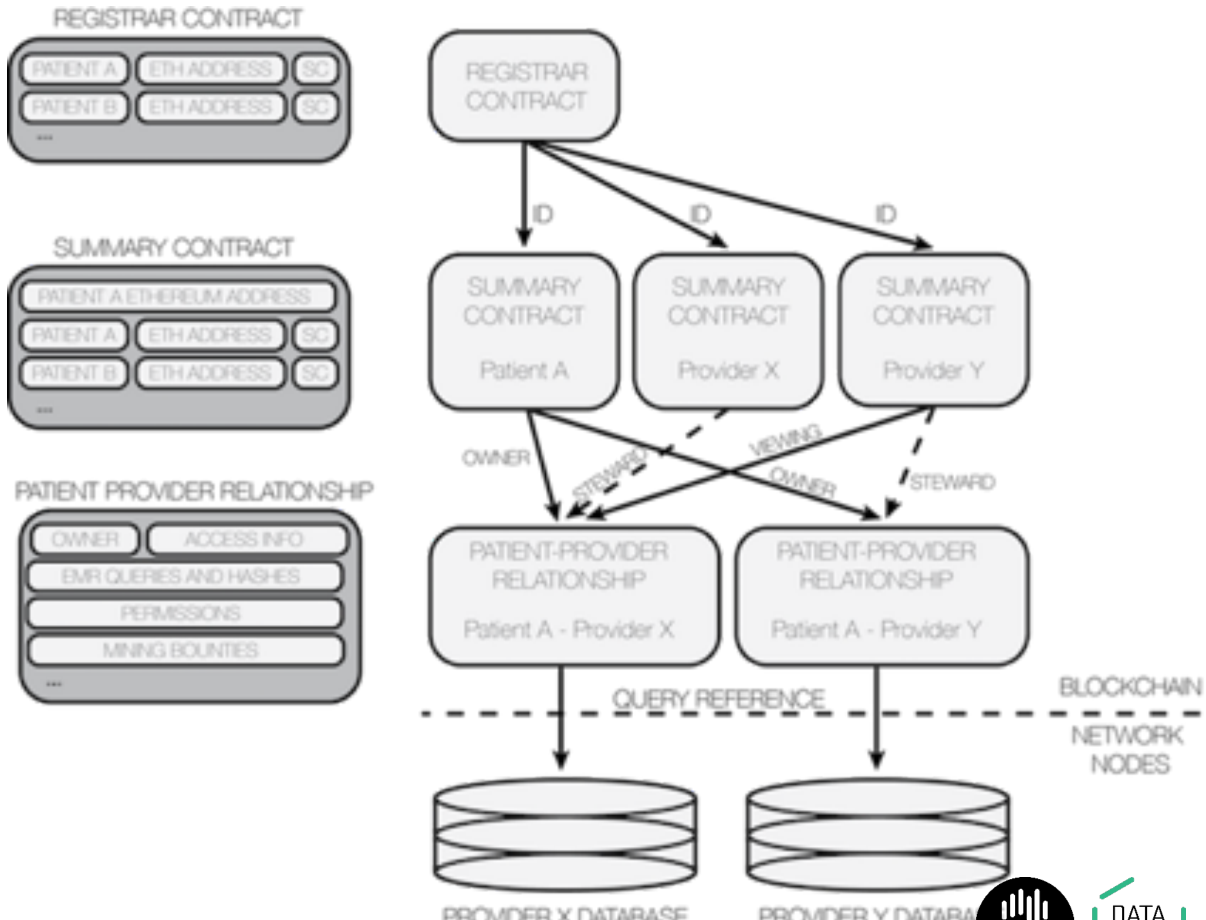
	Electronic health records
Multi-party	Required
Trusted authority	Decentralized
Centralized operation	Not required
Data immutability & Non-repudiation	Required
High performance	Not required
Data transparency & Confidentiality	Confidential
Sample Result	Conventional System

- **No low latency updates**
  - Most records do not change often
  - Large diagnostic image needs to be managed
- Due to privacy constraints, blockchain can not used to store patient records, even in encrypted form
- Conventional systems are used
  - Blockchain provides auxiliary service
    - Keep audit logs of accesses made to EHR
    - Ensure data integrity

Use Case 2: Electronic Health Records 3/4

MedRec

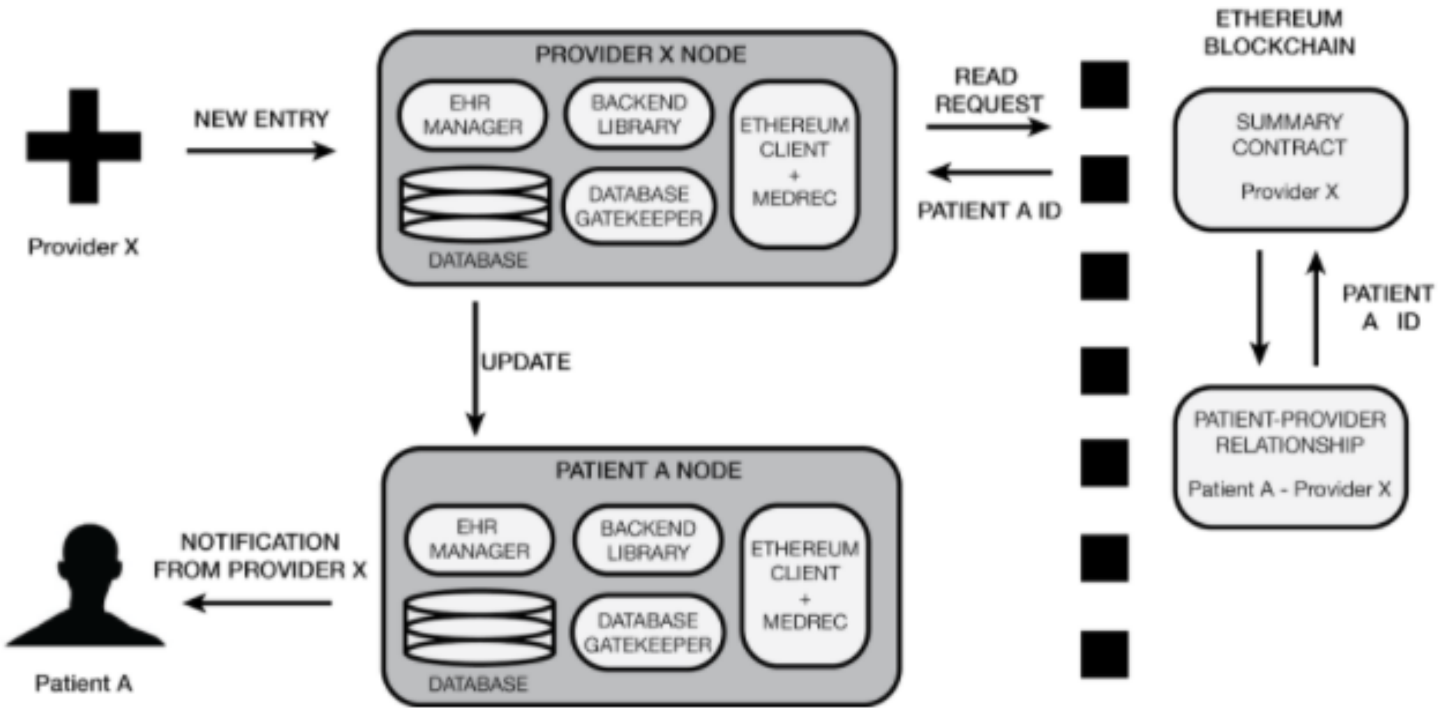
- Initiative to explore on blockchain architecture in contributing to secure and interoperable EHRs system



Use Case 2: Electronic Health Records 4/4

MedRec 2.0

- Improve scalability
  - Bypass the blockchain for patient notification
  - Restrict blockchain storage to creation and modification of identities and relationships



Use Case 3: Identity Management 1/2

- Individuals, organizations, devices and assets can be identified by many schemes
  - Passport, wedding certificates, serial number, registration certificate
- Conventionally, the **operations are centralized**
- Managed by **a trusted authority**
  - Set permissions and role for users to ensure they only access parts of the system relevant to them
  - Integrity is critical
- Complicated authorization
  - Requirement for delegated authorization
  - Requirement for dynamic revocation of authorizations
- Logs of system accesses are required to be able to audit
- **Read accesses** can be frequent, **updates to information** are less frequent
  - Some delay in propagating updates is acceptable

Use Case 3: Identity Management 2/2

	Identity
Multi-party	Required
Trusted authority	Not required
Centralized operation	Not required
Data immutability & Non-repudiation	Required
High performance	Not required
Data transparency & Confidentiality	Transparent
Sample Result	Blockchain

- Blockchain allows the roles, permission and privilege of users to be verified by the distributed peers
  - Remove the centralized administrator
  - Remove the centralized database
  - Ensure integrity of user identities, roles, and authorization
- Privacy is critical
  - Plaintext identity information is kept off-chain or encrypted on-chain

Use Case 4: Stock Market 1/2

- A place where stocks, bonds and securities are traded
- Inherently involves **multiple entities** to issue and trade stocks
- Conventionally implemented by a **centrally-controlled and maintained** register
  - Regulatory approval is required for the operation of stock market
  - Regulatory approval may be required for the trading of specific stocks
- Stock market is a natural **trusted authority**
- **Integrity, immutability and non-repudiation** is critical
  - Ensure high-value trades cannot be undone by any party
- **Transaction history** is important in providing evidence for trades and current stock holdings
- Typically have **a high-volume**, **extremely low-latency** price-setting mechanism
  - Settlement can have **high throughput** but **does not have extreme latency**

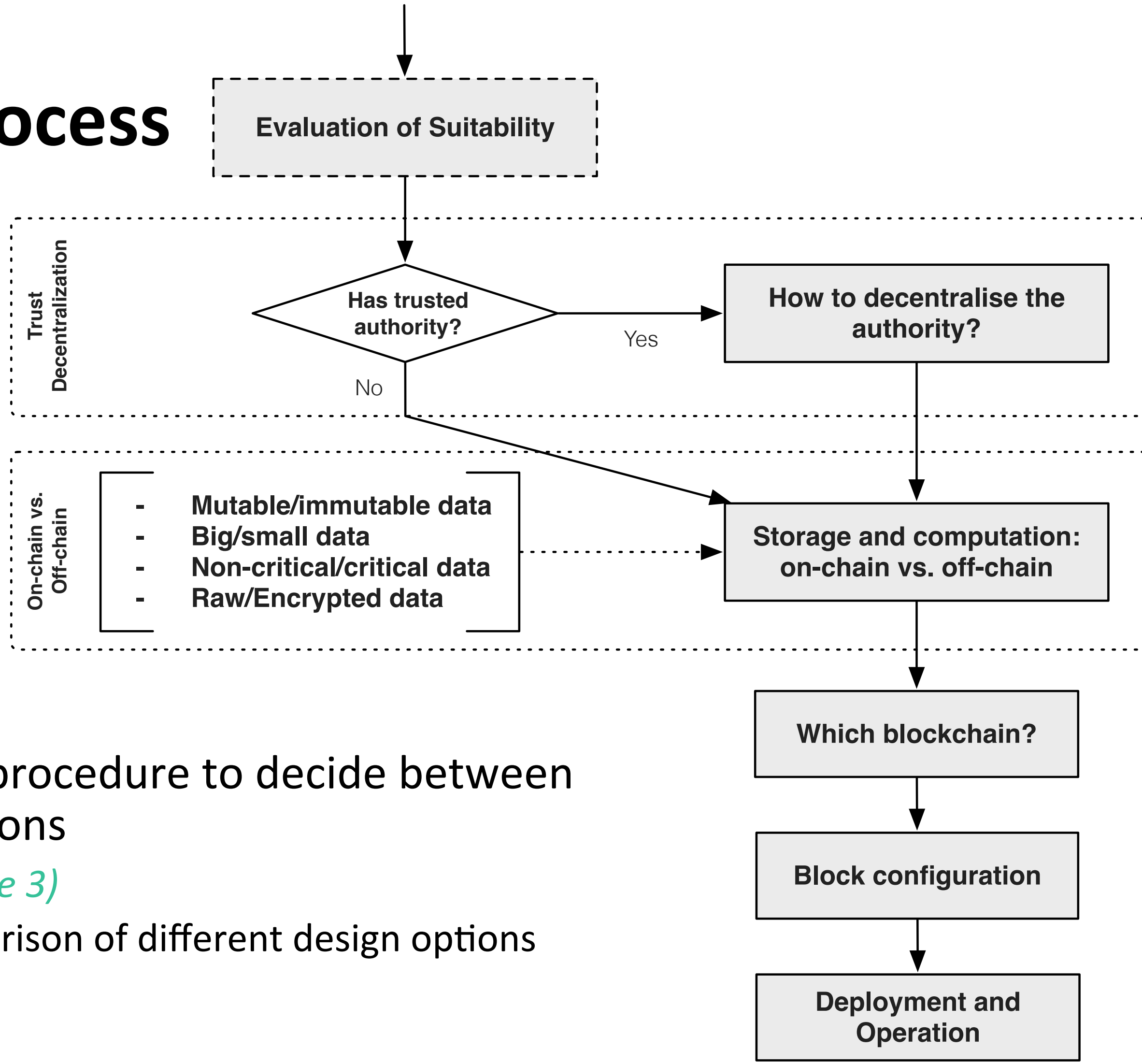
Use Case 4: Stock Market 2/2

	Stock market
Multi-party	Required
Trusted authority	Not required
Centralized operation	Not required
Data immutability & Non-repudiation	Required
High performance	Required
Data transparency & Confidentiality	Confidential
Sample Result	Conventional System

- Blockchain allows settlement using peer confirmation
  - Remove the centralized operation and authority
- **Data transparency is an issue**
  - All investors and market participants are exposed to blockchain participants
- Data immutability is crucial
  - Ensures no successful transaction can be tampered with by anyone
- Blockchain is **not highly suitable** for the operation of conventional regulated stock markets
  - Scalability issue
- NASDAQ offers Linq ledger for registration and settlement of securities

Design Process for Blockchain-based Systems

Design Process



- Every step is a procedure to decide between alternative options
  - Taxonomy *(Lecture 3)*
  - Systematic comparison of different design options

Trade-off Analysis 1/2

Design decision	Quality
Block size, Block frequency	Scalability
Consensus protocol	Security
Public/Consortium/Private blockchain	Cost efficiency
Data structure	Performance

- Design decision
  - Improve the performance of one quality attribute
  - May harm the performance of other quality attributes

Trade-off Analysis 2/2

Encrypting data before storing it on a blockchain

- Increase confidentiality
- Reduce performance, may harm transparency or independent auditability

Storing only a hash of data on-chain and keeping the contents off-chain

- Improve confidentiality and performance
- Partly undermine the benefit of blockchains in providing distributed trust
- Single point of failure reduces system availability and reliability

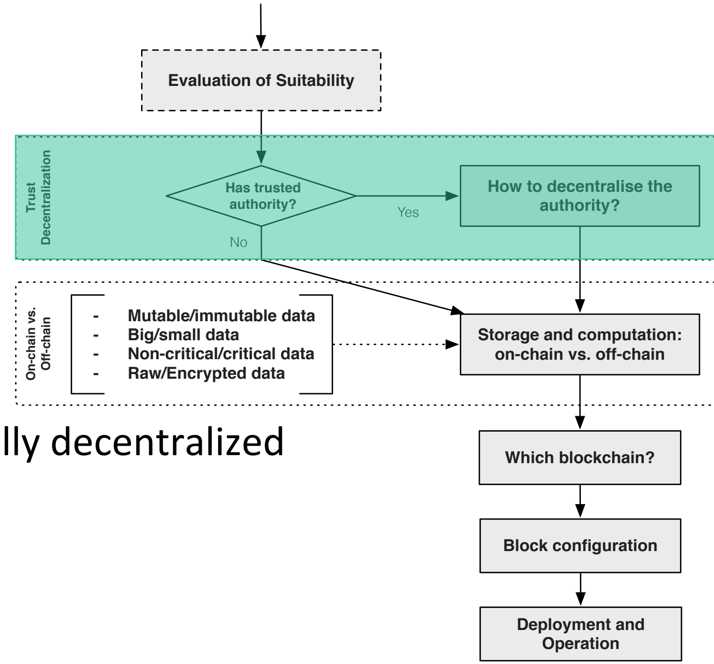
Using private blockchain instead of public blockchain

- Allow greater control over the admittance of processing nodes and transaction into the system
- Increase barriers to entry for participation, thus partly reduce some benefit of using blockchain

Higher number of confirmation blocks

- Increase confidence in integrity and durability of transaction
- Harm latency

Decentralisation



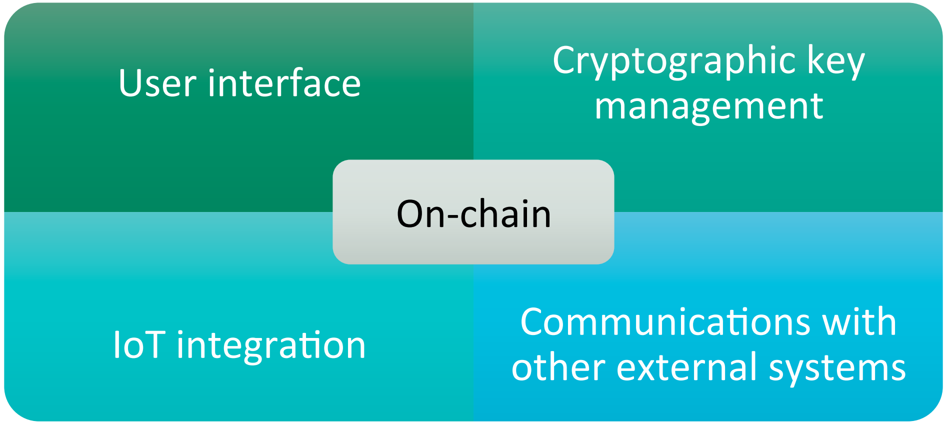
- Blockchain is used in scenarios
  - Where no single trusted authority is required
  - Where the trusted authorities can be decentralized or partially decentralized

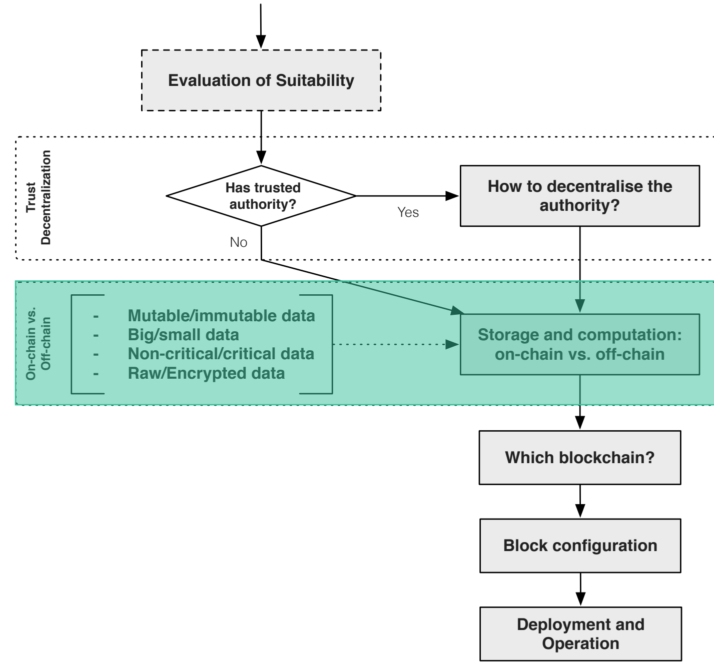
- Deployment and operation of system spectrum

Centralized monopolies → Central parties with a competition between parties → Services provided jointly by a consortia → Full open service provision in a peer-to-peer system

- Some components or functions are decentralized while others are centralized

On-chain vs. Off-chain





- Many kinds of data are better stored off-chain
  - Scalability reason—“Big” data
    - “Not tiny” data may be too large to store on blockchain
  - Confidentiality reason – private data
  - Dealing with legacy database

Design Decisions and Their Impact

Design Decision		Option	Impact			
			Fundamental properties	Cost efficiency	Performance	Flexibility
Data	On-chain	Embedded in transaction (Bitcoin)	⊕⊕⊕⊕	⊕	⊕	⊕⊕
		Embedded in transaction (Public Ethereum)		⊕⊕⊕⊕	⊕	⊕⊕⊕
		Smart contract variable (Public Ethereum)		⊕⊕	⊕⊕⊕	⊕
		Smart contract log event (Public Ethereum)		⊕⊕⊕	⊕⊕	⊕⊕
	Off-chain	Private / Third party cloud	⊕	~KB Negligible	⊕⊕⊕⊕	⊕⊕⊕⊕
		Peer-to-Peer system		⊕⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Computation	On-chain	Transaction constraints	⊕⊕⊕⊕	⊕	⊕	⊕
		Smart contract				
	Off-chain	Private / Third party cloud	⊕	⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕⊕

On-chain Data

Colour coin as On-chain auxiliary data
- Overlays on Bitcoin to represent real world assets

- A common practice of storing item data
  - Raw data off-chain
  - On-chain just meta-data, small critical data and hashes of the raw data
  - On-chain data not just for integration with external data

- Store data in Bitcoin
  - *OP_RETURN* (limited to 40 bytes)
  - Slow and costly

- Store data in Ethereum
  - Storing arbitrary data in transaction
    - Transaction size is limited by the maximum size of a block
    - Practically smaller transactions accepted by other users
  - Storing data in smart contract
    - As variable in a smart contract
    - As log event of smart contract
    - Variable is more efficient to manipulate, but less flexible due to the constraints of language

Cost Model

- Monetary cost of using public blockchains follows a different cost model than conventional software systems *(Second half of this lecture)*
- More expensive
- One-time cost for permanent storage
  - Partial refund of Ethereum

Off-chain Data Storage

- Concern the interaction between blockchain and data storage facilities

- Cloud storage
- Private cloud on the client's infrastructure
- Public storage provided by a third party
- Data replication is managed by the system or consumer

- Peer-to-peer data storage
- IPFS (InterPlanetary File System)
    - Free, but availability depends on IPFS server that hosts the data
- StorJ
- Data is replicated automatically, or based on the user behavior

Design Decisions and Their Impact

Design Decision		Option	Impact: Fundamental properties	Impact: Cost efficiency	Impact: Performance	Impact: Flexibility
Data	On-chain	Embedded in transaction (Bitcoin)	⊕⊕⊕⊕	⊕	⊕	⊕⊕
		Embedded in transaction (Public Ethereum)		⊕⊕⊕⊕	⊕	⊕⊕⊕
		Smart contract variable (Public Ethereum)		⊕⊕	⊕⊕⊕	⊕
		Smart contract log event (Public Ethereum)		⊕⊕⊕	⊕⊕	⊕⊕
	Off-chain	Private / Third party cloud	⊕	~KB Negligible	⊕⊕⊕⊕	⊕⊕⊕⊕
		Peer-to-Peer system		⊕⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Computation	On-chain	Transaction constraints	⊕⊕⊕⊕	⊕	⊕	⊕
		Smart contract				
	Off-chain	Private / Third party cloud	⊕	⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕⊕

Computation

- Different levels of expressiveness of on-chain computation
  - Bitcoin only allows simple scripts and conditions
    - Satisfied to transfer Bitcoin
  - Ethereum provides a Turing complete programming language
    - Not only perform conditional payments
    - Make modification to the working data in smart contract variables
  - DAML (Digital Asset Modelling Language)
    - More expressive than Bitcoin Script
    - Purposefully not Turing-complete: codify financial rights and obligations
      - In order to facilitate static analysis

Non-determinism

- Blocks impose an order on transactions Resolving non-determinism
- Otherwise might affect the execution results

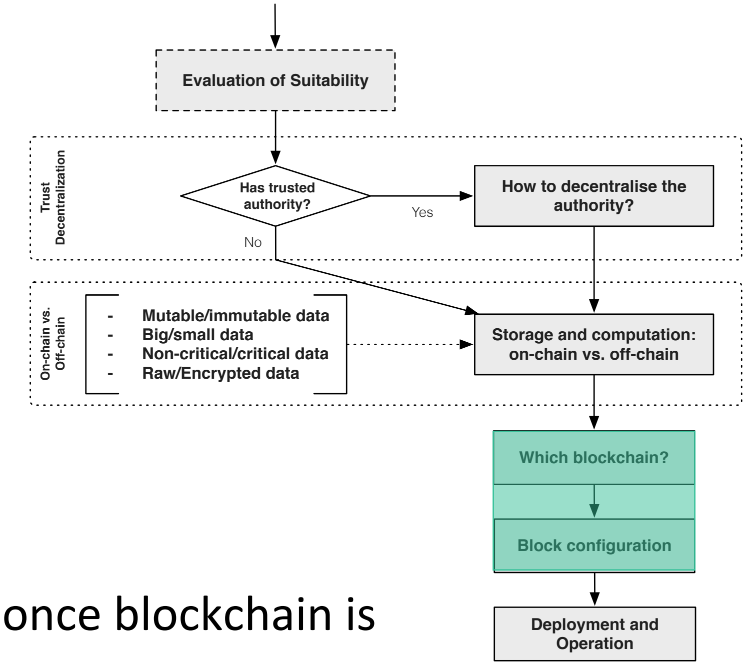
Digital Asset

- Benefit of on-chain computation
  - Inherent interoperability among the systems built on the same blockchain network
  - Neutrality of execution environment
  - Immutability of the program code once deployed

Impact of Other Components

- Other components in the broader system has impact on design decisions
- What's on-chain vs. what's off-chain
- Identity management *(One of the example use cases)*
- Supports systems where has a requirement to know individual human or system involved in transactions
- International payments have regulatory requirement to establish identity of participants
  - AML (Anti-Money Laundering) and CTF (Counter-Terrorism Financing)
- Real-world identity is not required from technical perspective
  - Bitcoin transacting agents are only cryptographically identified
  - International exchange can be performed without establishing real-world identity
- AML/CTF requirements are not obviated by the use of blockchain
- Off-chain protocol might be used to store identity information
  - Privacy and confidentiality can be a challenge when storing on-chain

Blockchain Selection

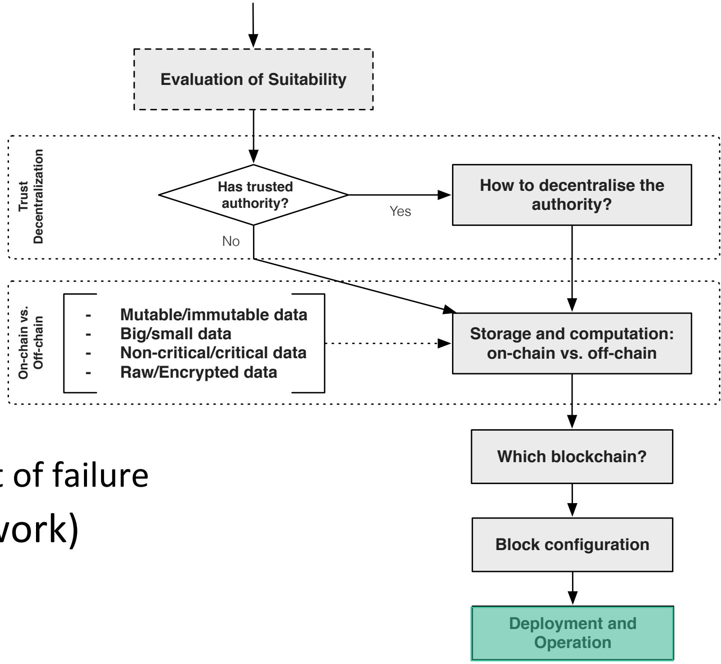


- Blockchain platform is selected
  - Requirement of the use case
  - Features of blockchain platform
  - Trade-off analysis

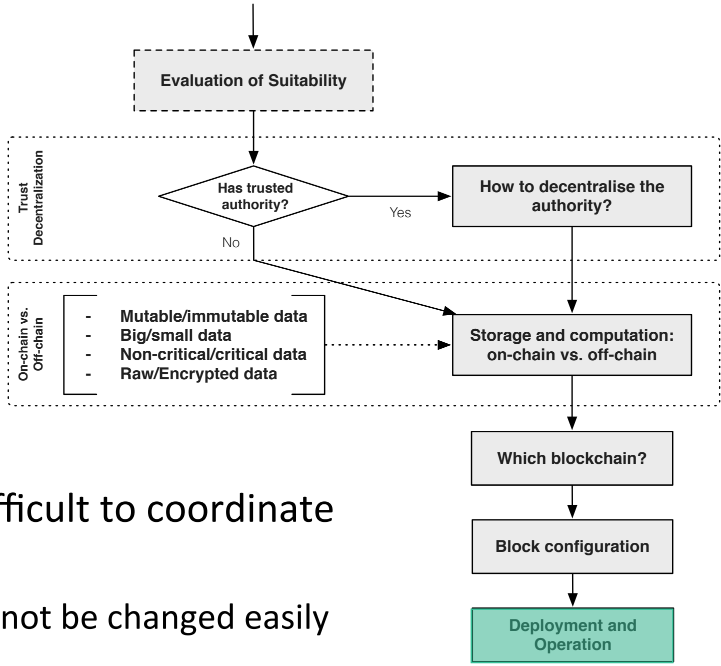
- Consensus protocol and other decisions are fixed once blockchain is selected
  - Hyperledger Fabric is an exception
    - Support pluggable implementations of various consensus protocols
  - Inter-block time is configurable for Proof-of-Work protocol
    - Through adjustments to the difficulty of mining

Deployment

- Deployment has impact on quality attributes
  - Deploying on cloud or using BaaS (Blockchain-as-a-Service)
    - Introduces the uncertainty of cloud infrastructure
    - Cloud provider becomes a trusted third-party and a single point of failure
  - Deploying a public blockchain on a VPN (virtual private network)
    - Becomes a private blockchain
    - Permissioned access controls provided at the network level
    - VPN introduces additional network latency overhead

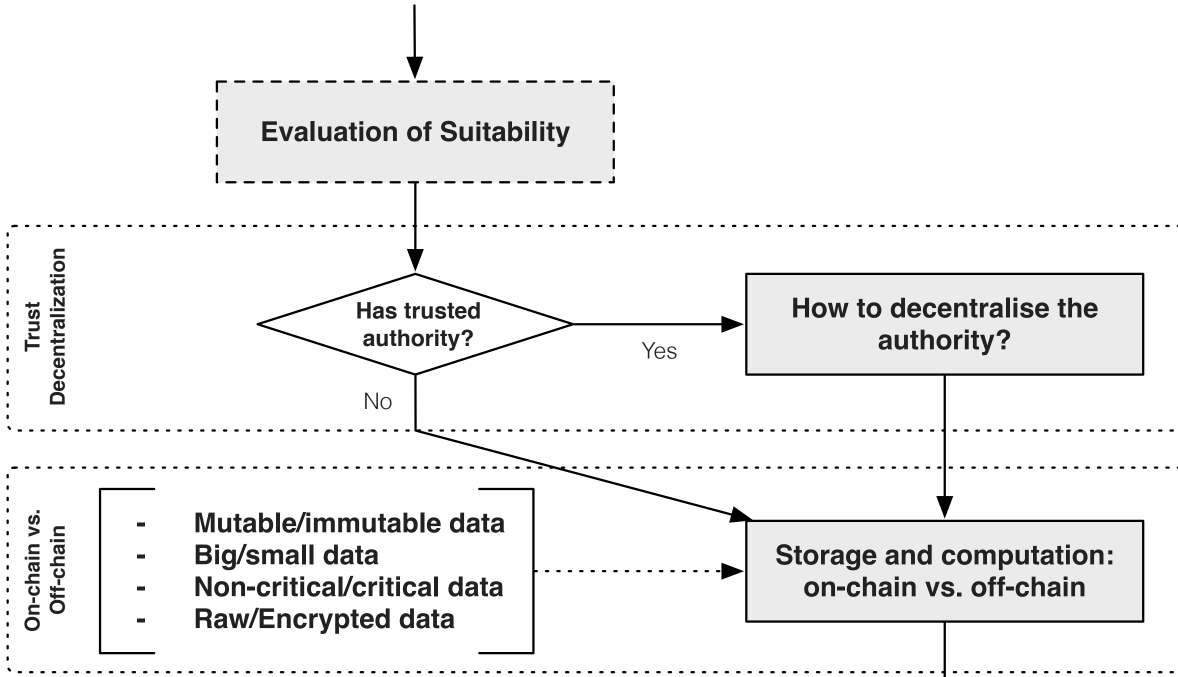


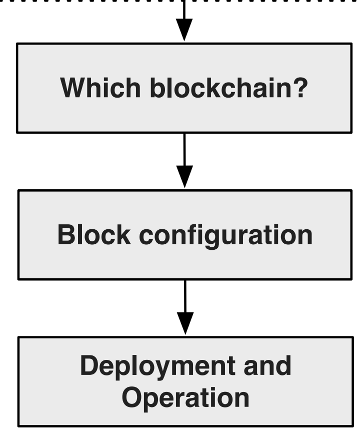
Operation



- Operation challenges
  - Blockchain-based systems can be harder to modify than conventional systems
  - Blockchain platform software running on multiple independently-operating nodes
  - Updating software can by physically and administratively difficult to coordinate
  - Blockchain is immutable by design, so cannot be updated
    - Trust is derived partly from the fact that the smart contract cannot be changed easily
- Governance
  - Changes may be made to correct defects, add features or migrate to new IT context
  - More like diplomacy in multi-party system with no single owner
  - Blockchain is ***NOT SUITABLE*** to a system that needs to change frequently
  - Learn from governance in open-source software
  - Governance concerns software development, deployment and operation
  - Immutable smart contract also simplify governance to some degree
    - Smart contract is available for execution while the blockchain operates normally

Summary





- Suitability needs to be evaluated first
  - Fundamental properties and limitations of blockchain
- Each step of the design process is a design decision
  - On-chain vs. off-chain
  - Subsidiary design decisions

COMP6452 Lecture 5.2: Cost

www.data61.csiro.au

Outline

- On-chain Data Cost
- Smart Contract Cost
- Cost Models
- Using Cost Model

Cost is important

- Monetary cost is as important for blockchain technologies as they are for conventional technologies
- Blockchain systems have different cost models
- Cost for storing too much data on-chain can explode quickly
- Blockchains enable distributed trust, but bring tradeoffs against execution cost and latency and throughput

Arbitrary Bytes in Unspendable Bitcoin Tx

Coinbase Transaction in Block

- Coinbase transaction mints new coins received by the miner who generates the block
- *Coinbase* parameter can contain arbitrary data
- Only the miner has access to this parameter

nSequence of a transaction

- Distinguish transactions from other Bitcoin transactions
- Presenting assets other than BTCs
- Every participants with the permission to submit transaction can set the value

Fake account address

- Sending a small amount of coins to the fake account
- The coin is lost forever
- Use 1-to-n transaction
- Minimal amount of funds to avoid denial of service attack

Conditional statements

- *OP_IF, OP_ELSE, OP_ENDIF*
- Clauses cannot be reached under any condition
- Extra overhead

OP_RETURN

- Official way to embed arbitrary data in a Bitcoin transaction
  - Returns immediately with an error
    - Included data is not interpreted as a script
  - Default Bitcoin client only relayed *OP_RETURN* transactions up to 80 bytes
    - Reduced to 40 bytes in 2014

- Storing 80 bytes of arbitrary data on the Bitcoin costs roughly US$0.459
  - Assuming a typical Bitcoin transaction with one input and one output (220 bytes)
  - The default transaction fee rate is 2×10^{-4}BTC/KB

- It is debatable whether Bitcoin should be used to record arbitrary data.

Exchange rates of US$7650 / BTC from 2 August 2018 (https://poloniex.com/exchange#usdt_btc)

Storing Data on Ethereum Transaction

80 bytes on Bitcoin costs US$0.459

- Theoretically allows storing arbitrary data of any size
- Storing 80 bytes of arbitrary data on Ethereum costs roughly US$0.22

- Every transaction has a fixed cost of 21,000 gas
  - Gas is the internal pricing for executing a transaction of storing data
- Every non-zero byte of data costs additional 68 gas
- Total cost of storing 80 bytes via transaction is 26,440 gas
  - Assuming all bytes are non-zero

Exchange rate of US$420 / ETH from 2 August 2018 (https: //poloniex.com/exchange#usdt_eth)
gas price of 2 × 10−9 ETH (2 Gwei) on Ethereum

Storing Data in Smart Contract

Storing 32 bytes of data
(simple types of Solidity are 32 bytes)

- Storing data as a variable in a smart contract
  - Cost is based on the number of *SSTORE* operations
  - 1 *SSTORE* operation that changes the data from zero to non-zero (20,000 gas)
  - Transaction as the carrier costs a base 21,000 gas
  - Data payload costs extra gas
    - Function signature and the actual data
  - Cost for creating the smart contract depending on its complexity
  - Total cost is > US$0.036 (20,000 + 21,000 + 32 × 68 gas)
  - Subsequent transactions to **update** data costs 5,000 gas (keeping the data as non-zero)
  - Cost of subsequent transactions is ~ US$0.024 (5000 + 21,000 + 32 × 68 gas)
  - Less flexible due to the constraints of Solidity on the value types and length
- Storing data as a log event in a smart contract
  - 1 log topic costs 375 gas
  - Every byte of data costs an extra 8 gas
  - Transaction as the carrier costs a base 21,000 gas
  - Total cost is ~US$0.018 (21,000 + 375 + 32 × 8 gas)

Smart Contract Cost

- Cost charged on transactions in relation to their complexity
  - Base cost for any transaction (21, 000 gas)
  - Variable components
    - Data attachments
    - Contract execution is charged per *bytecode* instruction
    - Additional cost for contract deployment

- *Gas*
  - All cost follows a fixed pricing table specified in the unit *gas*
  - Gas cost is converted to Ether
  - User-defined *gas price* factor
    - How much Ether-per-gas is the transaction creator willing to pay
    - Default value is the current market rate
      - average over previously included transactions

Gas Limit

- Block gas limit
- Sum of gas used by the set of transactions included in a given block cannot exceed this limit
- Set by the miners
- Defined in terms of gas usage
  - Cannot be influenced by variations the user has power over
  - For example, underbidding the market price
- Making it a limit of complexity for new blocks
- An upper bound to throughput scalability
  - Cost of transactions vary
  - Non-trivial to understand how the bound relates to transaction throughput

- https://ethgasstation.info/index.php

Cost Model

- **Cost Model of Blockchain Infrastructure** (Ethereum)
  - Turing complete language to implement business logic
- **Cost Model of Amazon Web Services** (Amazon SWF)
  - Dedicated to process execution
    - Commonly-used workflow patterns and messaging patterns
  - AWS is a leading commercial cloud computing provider

(Use the execution of an instance of a business process model as sample application)

Ethereum Transaction 1/2

- 3 types of transactions
  - Financial transfer, message call and contract creation
  - Basic elements: `from`, `to`, `gasLimit`, `value and data`

- Financial transaction
  - `From/to` sender/recipient of the transaction
  - `Value` the amount transferred
  - `Data` (optional) data in arbitrary form
    - E.g. XML, pictures
    - Fee for a transaction covers the cost for storing the data permanently

Ethereum Transaction 2/2

- Message call transaction
  - Invoke a function of a contract
  - `From/to` sender/recipient of the transaction
  - `Value` (optional)
  - `Data` the method to be invoked and the parameters
  - `gasLimit` maximum gas can be used in this transaction
    - Gas is paid for each executed bytecode instruction

Cost of executing business process

- Contract creation transaction
  - `to NULL`
  - `Data` the contract bytecode
  - `Value` (optional)

Cost of deploying business process

Contract Creation Cost 1/2

- Contract creation transaction
  - `Data` compiled bytecode
    - Permanent storage incurs cost
  - `Value` (optional)
    - Endowment upon initialization
  - Ethereum address is assigned to it
    - Calculated with a deterministic function depending only on the creator's Ethereum account

$$C_{pload} = \textit{payload (in bytes)} \times C_{gas/byte}$$

- Cost of payload for contract bytecode is 200 gas per byte
- Cost of payload for data in a financial transaction/message call is 68 gas per non-zero byte and 4 gas per zero byte

$$C_{create} = C_{tx} + C_{addr} + C_{pload} + C_{fndef}$$

- C_{tx}: 21,000 base gas for transaction itself
- C_{addr}: 32, 000 for allocating address
- C_{fndef}: consumed by the the function definition

Contract Creation Cost 2/2

- Contract can be created by another contract
  - Not via transaction
  - Cheaper Without C_{tx}

$$C_{create}' = C_{addr} + C_{pload} + C_{fndef}$$

Contract Execution Cost

- Function call cost

$$C_{execute} = C_{tx} + C_{pload} + C_{fnexe}$$

- C_{tx}: 21,000 base gas for transaction itself
- C_{pload}: cost of data payload
- C_{fnexe} : consumed by the opcode present during the function execution

Gas Cost → Ether → Another Currency

$$C_{\$} = C_{Gas} \times \text{gasPrice} \times 10^{-18} \times EXC_{ETH2CUR}$$

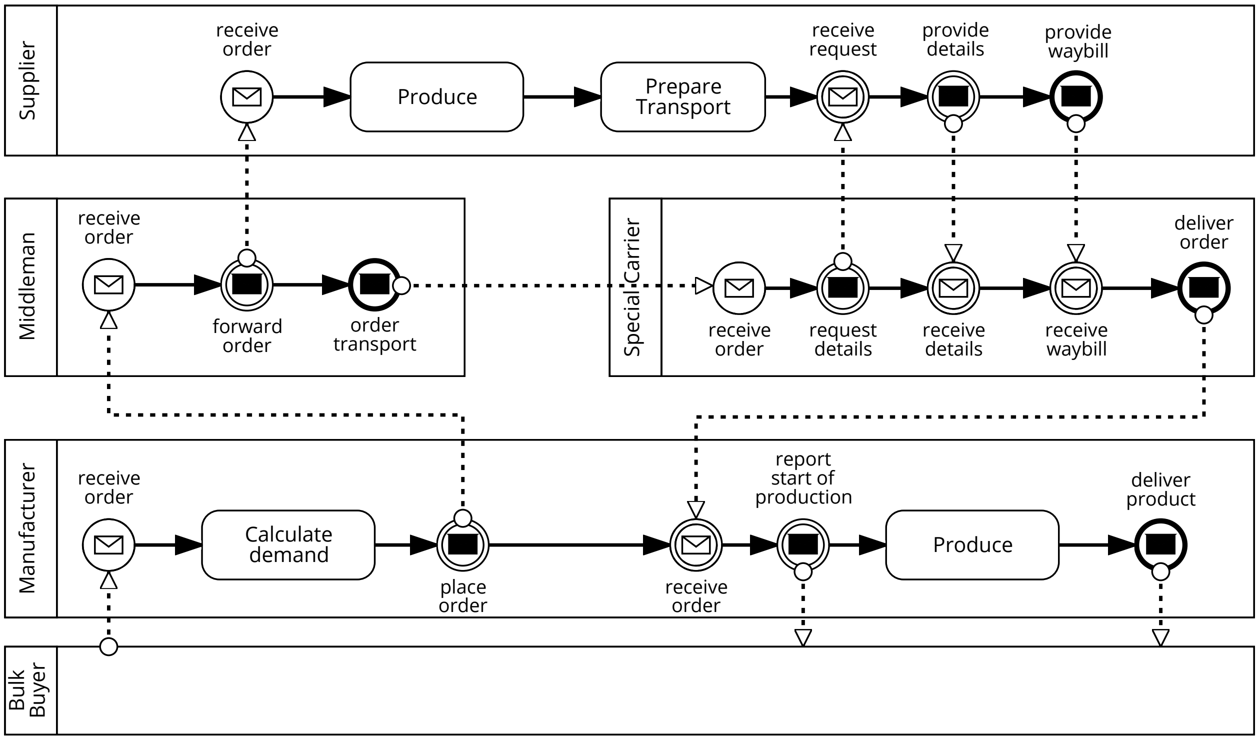
- C_{Gas} *:* cost in gas
- Gas price in *wei* (10^{-18} *Ether*)
- $EXC_{ETH2CUR}$ *:* Exchange Rate

https://coinmarketcap.com/currencies/ethereum/

https://fx-rate.net/ETH/USD/

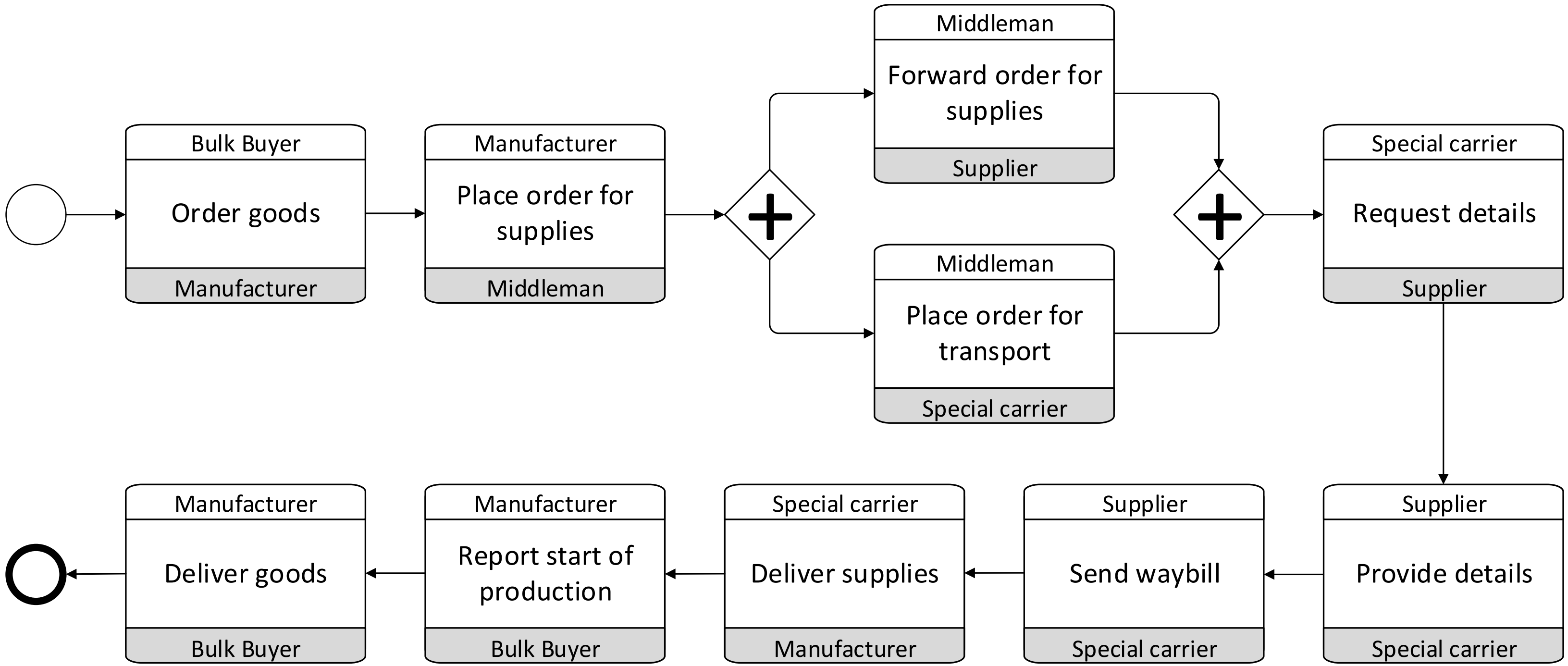
Cost of Interaction Component 1/5

Supply Chain Process in BPMN Orchestration



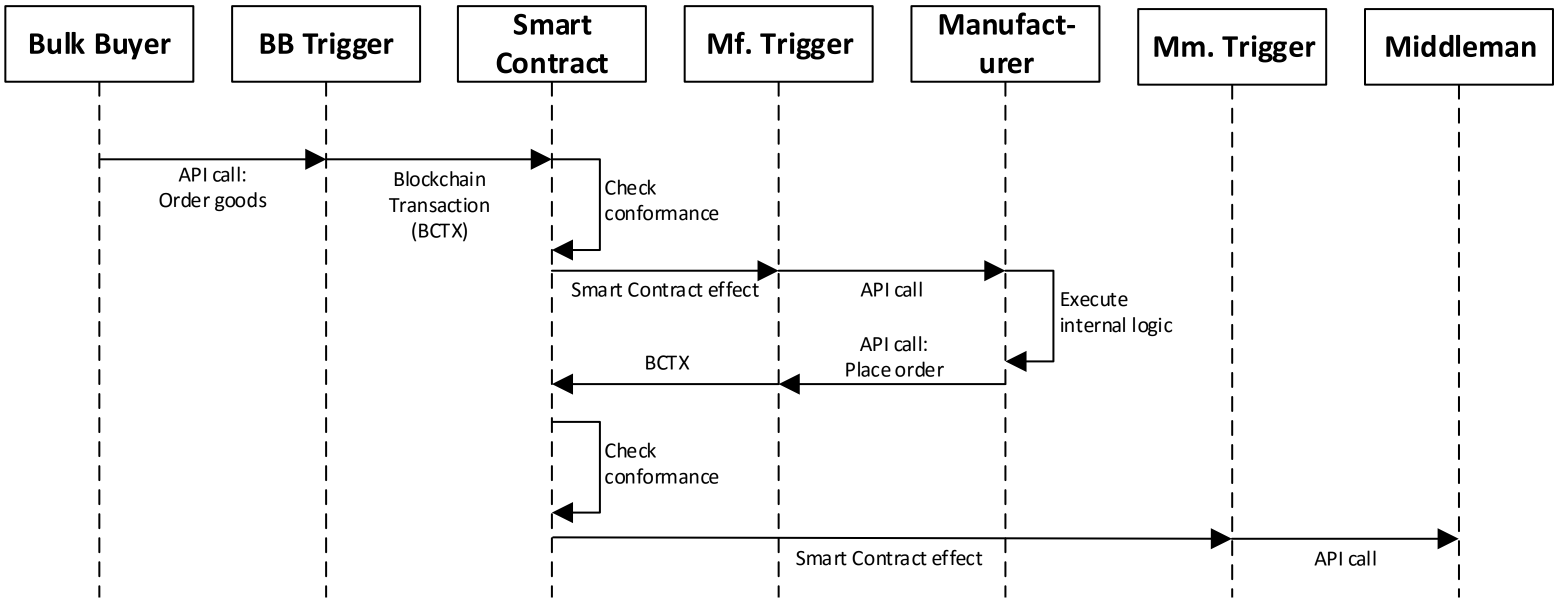
Cost of Interaction Component 2/5

Supply Chain Process In BPMN Choreography



Cost of Interaction Component 3/5

Sequence Diagram of the First Two Tasks



- Interaction between internal process implementation, triggers, and process instance smart contract

Cost of Interaction Component 4/5

Blockchain ⟺ Interface component ⟺ Enterprise system

- Assuming the interface component is running on AWS

$$C_{comp} = EC2_{price}(ec2_t) \times time$$

- $EC2_t$: the set of all available VM types in AWS $ec2_t \in EC2_t$
- Capacity of VM
  - $TP_{bc} : EC2_t \rightarrow R$
- Determines VM type based on workload
  - $f_{bc} : (TP_{bc}, WL_{bc}) \rightarrow EC2_t$
- Cost of running a VM of this type per billing time unit (BTU)
  - $EC2_{price} : EC2_t \rightarrow R$

Cost of Interaction Component 5/5

- Interface component operates a full node
  - Synchronize the blockchain if the VM is not constantly online
  - Required duration includes the time of synchronization
- Ethereum clients allows fast synchronization (“fast” flag)
  - Downloading transaction receipts instead of the full set of known blocks
    - Shows that the transactions happened but
    - NOT show the results of the smart contract function execution
    - Less evidence for integrity
    - Only be done when downloading the blockchain from scratch
      - Takes on the order of hours to days for public Ethereum blockchain

Cost Model

- Cost Model of Blockchain Infrastructure
- **Cost Model of Amazon Web Services (SWF)**

(Use the execution of an instance of a business process model as sample application)

Amazon SWF

- Simple Workflow Service (SWF) provided by AWS
  - Representative for cloud-based business process execution
  - Clear mapping to the process model
  - Tiered pricing model
    - More usage results in cheaper cost per unit

Workflow Executions

A workflow is a set of tasks executed in a certain order (sometimes with a set of conditional flows or loops). Each time that a workflow is executed, it is considered a distinct workflow execution. You pay for workflow executions when you start them (i.e. their first task becomes available for application hosts to execute) and for each 24-hour period until they are completed. The first 24 hours of workflow execution are free.

Start a Workflow Execution

Region: US East (Ohio)

- $0.00 for first 1,000 workflow executions
- $0.0001 per workflow execution above the free tier

Data Transfer **

Region: US East (Ohio)

Data Transferred	Pricing
Data Transfer IN	
All data transfer in	$0.00 per GB
Data Transfer OUT* **	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

SWF Cost Model

- Main elements: workflow, actor, task, and signal
- Workflow: A collection of activities in a specified sequence
  - An instance of business process
- Actor: Play participant roles from the business process
- Activity (task): Schedule a notification to the appropriate actor to proceed with the next activity
- Decision (task):
  - Determine whether the current state of execution conforms to the workflow
  - Determine which activity to execute next
- Signal: External triggered event to a currently executing workflow

Element Mapping

Business Process	Blockchain	Amazon SWF
Process instance	Instance of Smart Contract	Workflow
Conformance checking	Contract execution	Decision task
Activity	Contract execution	Activity task
Incoming message	Transaction	Signal
Outgoing message	Entry in contract event log	Notification

- Conformance checking is a technique in process mining
  - compare an existing process model with an event log produced by the process model
  - check if what happened in reality conforms to the process model and can be used at runtime

Base Cost of Workflow Instance

$$C_{wf} = \#wf \times SWF_{wf}$$

- *#wf :* Number of instances
- SWF_{wf} SWF cost of starting a workflow execution

Cost of Scheduling Tasks

$$C_{task} = (\ \#actTask + \#decTask) \times SWF_{task}$$

- *#actTask :* number of activity tasks
- *#decTask :* number of decision tasks
- SWF_{task} : price per task
- No. SWF activity tasks = No. activities in a business process
- No. SWF decision tasks = No. activities in a business process
- Schedule a decision task every time receiving a signal (completion of a activity)
- Process initiation creates a decision task to instruct the workflow to wait for the first signal

Cost of Signals

$$C_{sig} = \#signals \times SWF_{signal}$$

- *#signals :* number of signals
- SWF_{signal} *:* price per signal

Cost of Data Retention and Transfer

$$C_{ret} = (execT + retT) \times SWF_{ret}$$

$$C_{dat} = payload \times SWF_{data}$$

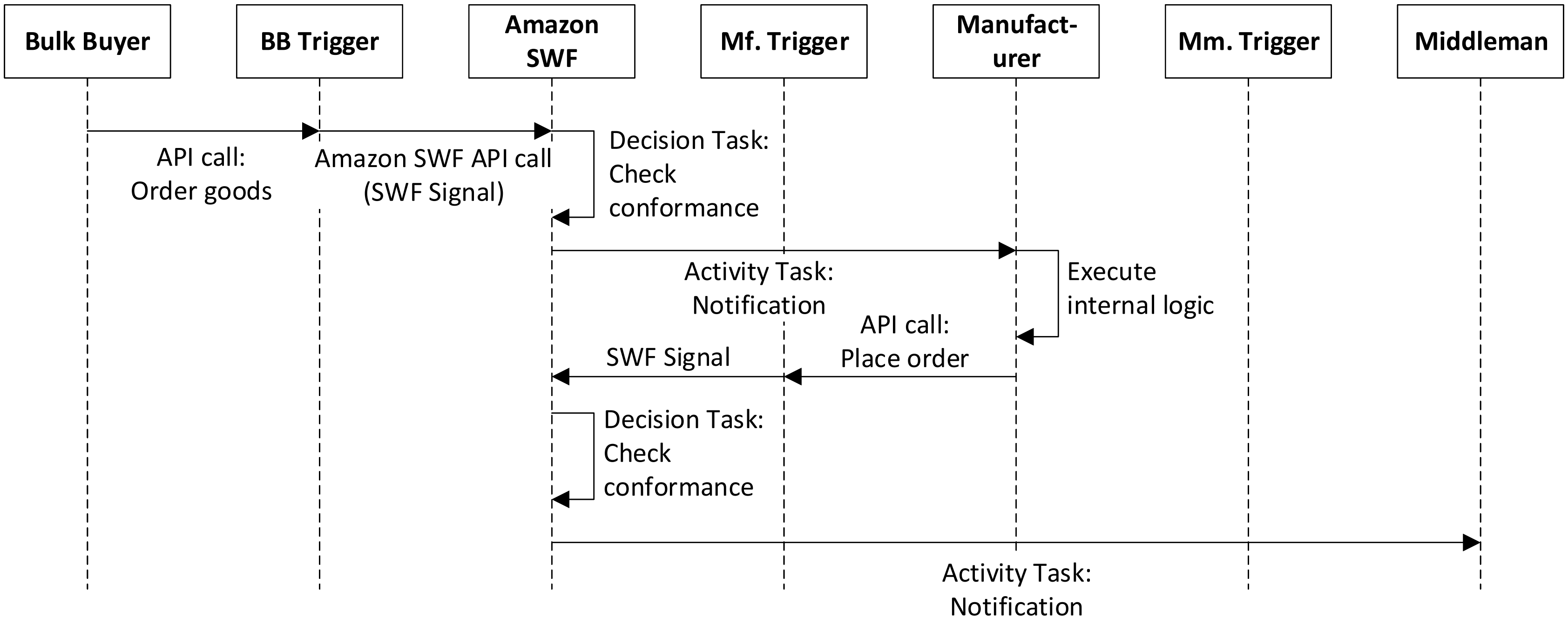
- *retT:* user-specified duration for generated data being retained
  - Charged for storage per 24 hours
- *execT :* Workflow execution time
  - Charged per 24 hours at the same rate as data retention cost
- SWF_{ret} : SWF cost rate
- *Payload:* inwards and outwards data size
- SWF_{data} *:* price per data unit

Coordination Cost

$$C_{swf} = C_{wf} + C_{task} + C_{sig} + C_{ret} + C_{dat}$$

Cost of Interaction Component 1/2

Using Amazon SWF



- Task execution requires actor running a *Amazon SWF worker* module
  - On AWS EC2 or internal infrastructure
  - Execute both decision task and activity task

Cost of Interaction Component 2/2

- Cost of VMs
  - Triggers and Amazon SWF workers
- Throughput per VM type

$$\mathrm{TP}_{swf} : EC2_t \rightarrow \mathrm{IR}$$

- VM type depends on capacity of VM types and the workload
  - The throughput values are different from the ones for blockchain triggers

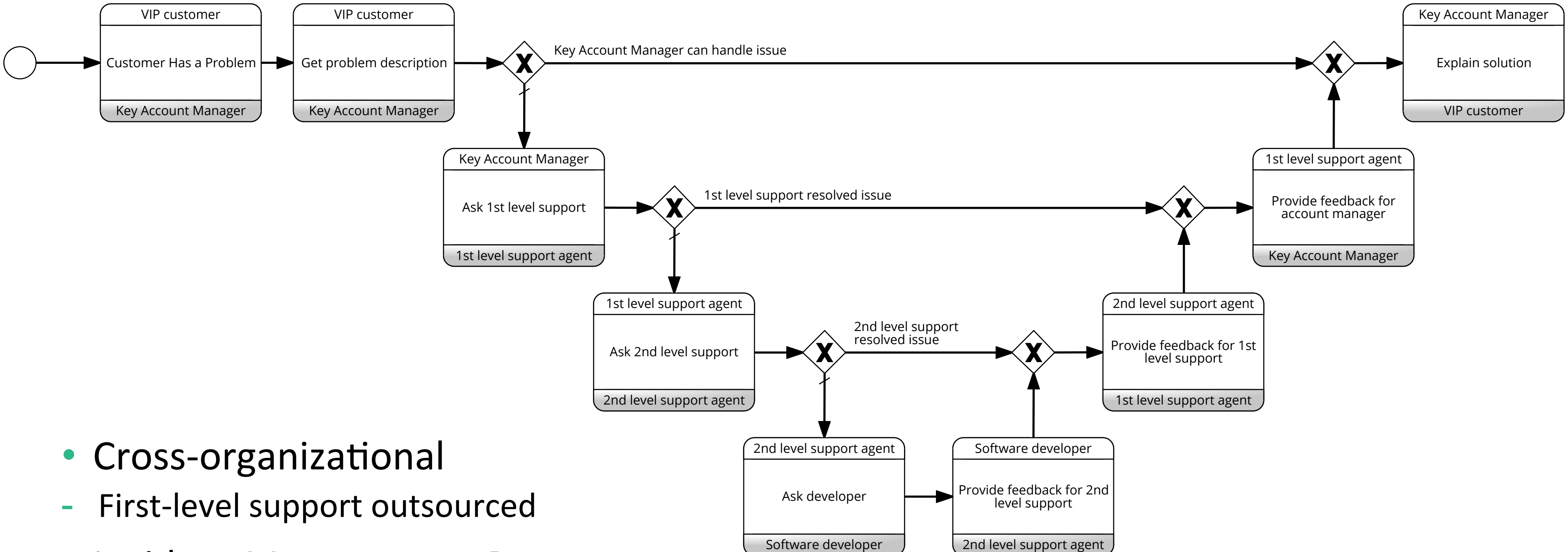
$$f_{swf} : (\mathrm{TP}_{swf}, WL_{swf}) \rightarrow EC2_t$$

- Cost of running the VMs for the time needed

$$\mathrm{C}_{comp} = EC2_{price}(ec2_t) \times time$$

- Minimum requirement is one VM to host the trigger and SWF worker
- Preferable setup is that each participant at least one VM
  - Host their own trigger and worker

Evaluating Cost Model



- Cross-organizational
  - First-level support outsourced
- Incident Management Process
  - 9 tasks
  - 6 gateways
  - 4 conforming traces

AWS VM Throughput Benchmark

VM Types	vCPU Specifications	Memory (GiB)
t2.small	1 Intel Xeon E5-2676 2.40 Ghz v3 w/ Turbo up to 3.3 Ghz	2
m3.medium	1 Intel Xeon E5-2670 2.50 GHz v2 (Ivy Bridge) Processors	3.75
m3.large	2 Intel Xeon E5-2670 2.50 GHz v2 (Ivy Bridge) Processors	7.5
m3.xlarge	4 Intel Xeon E5-2670 2.60 GHz v2 (Ivy Bridge) Processors	15

Decision limits

Decision	Bucket size	Refill rate / s
`RequestCancelExternalWorkflowExecution`	100	10
`ScheduleActivityTask`	500	100
`SignalExternalWorkflowExecution`	500	10
`StartChildWorkflowExecution`	500	2
`StartTimer`	1000	142

- AWS EC2 VM types and specification
- Throughput

Metrics	Blockchain			Amazon SWF	
	t2.small	m3.medium	m3.large	m3.medium (default)	m3.medium (incr. limit)
Transactions or Signals	13,580	7,336	20,104	73,871	152,404
Network In (MB)	102	114	128	138	168
Network Out (MB)	195	131	278	353	376
Duration (sec)	3,610	3,605	3,604	3,605	3,605
Average Tx/sec or Average signal/sec	3.8	2.0	5.6	20	42

Incident Management Blockchain Cost

- 32 process instances with a total 256 transactions
- Deployment of factory contract costs 0.032 Ether（One-time cost）
- Each run with data transformation costs 0.035 Ether

- Total cost is approx. US$1.34
- Exchange rate is US$420 / ETH
- Gas price of *2 Gwei*

Incident Management Amazon SWF Cost

- EC2 *t2.micro* VM for trigger and SWF task worker
- Process instances executed in sequence

- US$0.92 for 1,000 process instances
- US$0.0009 per instance
  - Data retention is 1 day
- US$0.0027 per instance
  - Data retention is 365 days

- Cost breakdown

Element	elements in experiment	Unit cost (US$)	Total cost (US$)
Decision Task	15,000	0.000025	0.375
Activity Task	7,000	0.000025	0.175
Signal	7,000	0.000025	0.175
Workflow	1,000	0.0001	0.1
Retention (24h)	1,000	0.000005	0.005
Execution Time (24h)	1,000	0.000005	0.005
Data Transfer	1	0.09	0.09

Comparison

- Cost on blockchain is three orders of magnitude higher than on Amazon SWF
  - Excluding the one-time factory contract deployment
- Blockchain stores the result permanently
  - As long as the blockchain is in existence
- Ongoing cost for data storage on Amazon SWF
  - Store for 243,863 days (approx. 668 years) to reach break-even

Volatility of Cryptocurrency

- Sensitive to the volatility of the exchange rate

Costs	Ethereum (in Ether)	Exchange Rate (in US$)				
		0.10	1.00	10.00	100.00	1,000.00
Incident Management (contract deployment)	0.0032	0.00032	0.0032	0.032	0.320	3.20
Incident Management (per process instance)	0.00347	0.000347	0.00347	0.0347	0.347	3.47

- Comparison
  - Exchange rate
  - Retention rate

Costs	SWF cost (in US$)	SWF vs Blockchain cost comparison in ratio with different exchange rates (ratio < 1 means Blockchain is cheaper)					Break-even rate (in US$)
		\$0.10	\$1.00	\$10.00	\$100.00	\$1,000.00	
Incident (24 hours)	0.000925	0.375 (0)	3.751 (+1)	37.51 (+2)	375.14 (+3)	3,751.35 (+4)	0.27
Incident (99 years)	0.181595	0.002 (-3)	0.019 (-2)	0.191 (-1)	1.91 (0)	19.11 (+1)	52.33

Why Blockchain

- Blockchain provides trusted storage and execution environment
  - No trust in any single third-party
- Conventionally participants need to jointly agree on a mutually-trusted third party
  - E.g. AWS (for confidentiality and truthful execution)
  - The party controlling the Amazon SWF account

Co-opetition
- Organizations cooperate for cases to achieve business goals
- Compete in other cases

- Public blockchain inherently supports payment and escrow
  - Sending cryptocurrency with existing messages would not incur additional cost
    - Due to a flat fee structure
  - Offset the premium cost of distrust
    - Commercial escrow service charge 0.5% to 3.25%
    - Lower the cost of process executions involving monetary transaction

Cost vs. Maintainability

- Deployment methods impact cost and non-functional properties
  - (1) One smart contract with two functions
  - (2) Two smaller contracts, each implementing one function
    - One contract acts as an entry point
- The first has lower deployment cost
  - (2) needs to pay C_{tx} and C_{adr} twice
  - The payload of contracts in (2) is higher, as there are header bytes in the payload
- (1) is not as maintainable as (2)
  - One function needs to be modified
    - (1): updated contract needs to be redeployed as a whole
    - (2): only one contract is redeployed
    - (1): the triggers need to be updated with the new address
    - (2): might be avoided

Cost vs. Scalability of Triggers

- Additional resources are needed to accommodate increasing workload
  - Vertical scaling (bigger VM)
  - Horizontal scaling (more VMs)
- Blockchain nodes can scale vertically
  - Horizontal scaling has complication
    - Easy to add additional VMs into the network
    - Using one account from multiple VMs may lead to double-spending
    - Using different accounts on different VMs
      - Maintainability issues and increase storage costs
- SWF can scale both horizontally and vertically
  - Vertical scaling by choosing larger VM
  - Horizontal scaling by adding new VMs (registering it with SWF)
    - SWF then acts as load balancer
      - Distributing requests to multiple VMs

Summary

- Cost of basic compute and storage on public blockchain have different cost model than conventional cloud

- Public Ethereum and Amazon SWF are compared using business process
- Construct and benchmark cost model for both infrastructures
- Cost on public Ethereum blockchain is three orders of magnitude higher than on Amazon SWF

- Cost model incorporates exchange rate is important
- Given the high volatility of the exchange rate

- Cost is often in tradeoff with other non-functional properties
- Maintainability and Scalability

Course Outline – next two weeks

Week	Date	Lecturer	Lecture Topic	Relevant Book Chapters	Notes
5th	18 Mar	Sherry Xu	NPFs 1	6. Design Process for Applications on Blockchain 9. Cost	
6th	25 Mar	Mark Stapes	NPFs 2	10. Performance	Mid-term Exam (1 hour)
7th	1 Apr	Mark Stapes	NPFs 3	11. Dependability and security	Assignment 2 out (Monday before lecture)

THANK YOU

Xiwei Xu | Senior Research Scientist

Architecture & Analytics Platforms (AAP) team

t +61 2 9490 5664

e xiwei.xu@data61.csiro.au

w www.data61.csiro.au/

www.data61.csiro.au