SCIENCE
PASSION
TECHNOLOGY

Data Integration and Analysis
07 Data Provenane and Blockchain

Matthias Boehm

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

Last update: Nov 19, 2021

ISDS

Announcements/Org

- **#1 Video Recording**
  - Link in **TUbe** & **TeachCenter** (lectures will be public)
  - Optional attendance (independent of COVID)
  - Virtual lectures (recorded) until end of year
    https://tugraz.webex.com/meet/m.boehm

Agenda

- **Motivation and Terminology**
- **Data Provenance**
- **Blockchain Fundamentals**

Data Provenance
Motivation and Terminology

Excursus: FAIR Data Principles

[https://www.go-fair.org/fair-principles/]

- **#1 Findable**
  - Metadata and data have globally unique **persistent identifiers**
  - Data describes w/ rich **meta data**; registered/indexes and searchable
- **#2 Accessible**
  - Metadata and data retrievable via open, free and universal **comm protocols**
  - Metadata accessible even when data no longer available
- **#3 Interoperable**
  - Metadata and data use a formal, **accessible, and broadly applicable format**
  - Metadata and data use FAIR vocabularies and qualified references
- **#4 Reusable**
  - Metadata and data described with plurality of accurate and relevant attributes
  - Clear license, **associated with provenance**, meets community standards

Terminology

- **Data Provenance**
  - Track and understand data origins and transformations of data (**where?**, **when?**, **who?**, **why?**, **how?**)

  - Contains meta data, context, and modifications (transform, enrichment)
  - Synonyms: **data lineage**, **data pedigree**
- **Data Catalogs** (curation/**governance**)
  - Directory of datasets including data provenance (meta data, artifacts)
  - Raw/original, curated datasets, derived data products
- **Blockchain**
  - Data structure logging transactions in **verifiable** and **permanent way**

Applications and Goals

a) High-Level Goals

- #1 **Versioning and Reproducibility** (analogy experiments)
- #2 **Explainability, Interpretability, Verification**

b) Low-Level Goals

- #3 **Full and Partial Reuse of Intermediates**
- #4 **Incremental Maintenance of MatViews, Models, etc**
- #5 **Tape/log of Executed Operations** → Auto Differentiation
- #6 **Recomputation for Caching / Fault Tolerance**
- #7 **Debugging via Lineage Query Processing**

Data Provenance

Overview Data Provenance

- **Def Data Provenance**
  - Information about the **origin** and **creation process** of data
- **Example**
  - Debugging suspicious query results

```
SELECT Customer, sum(O.Quantity*P.Price)
  FROM Orders O, Products P
  WHERE O.PID = P.PID
  GROUP BY Customer
```

Customer	Sum
A	7620
B	120
C	130
D	75

OID	Customer	Date	Quantity	PID
1	A	2019-06-22	3	2
2	B	2019-06-22	1	3
3	A	2019-06-22	101	4
4	C	2019-06-23	2	2
5	D	2019-06-23	1	4
6	C	2019-06-23	1	1

PID	Product	Price
1	X	100
2	Y	15
4	Z	75
3	W	120

Overview Data Provenance, cont.

- **An Abstract View**
  - **Data:** schema, structure → data items
  - **Data composition** (granularity): attribute, tuple, relation
  - **Transformation:** consumes inputs, produces outputs
  - **Hierarchical transformations:** query w/ views, query block, operators
  - **Additional:** env context (OS, libraries, env variables, state), users

[Boris Glavic: CS595 Data Provenance – Introduction to Data Provenance, **Illinois Institute of Technology, 2012**]

- **Goal: Tracing of Derived Results**
  - Inputs and parameters
  - Steps involved in creating the result
  - → Store and query data & provenance
  - General Data Protection Regulation (**GDPR**)?

[Zachary G. Ives: Data Provenance: Challenges, Benefits, Research, **NIH Webinar 2016**]


```
1. Read file1
2. Sort rows
3. Compute median
4. Write to file2
```

Prov.

Classification of Data Provenance

- **Overview**
  - Base query Q(D) = O with database D = $\{R_1, \ldots, R_n\}$
  - **Forward lineage query:** $L_f(R_i'', O')$ from subset of input relation to output
  - **Backward lineage query:** $L_b(O', R_i)$ from subset of outputs to base tables
- **#1 Lazy Lineage Query Evaluation**
  - Rewrite (**invert**) lineage queries as relational queries over input relations
  - No runtime overhead but slow lineage query processing
- **#2 Eager Lineage Query Evaluation**
  - Materialize **annotations** (data/transforms) during base query evaluation
  - Runtime overhead but fast lineage query processing
  - Lineage capture: **Logical** (relational) vs **physical** (instrumented physical ops)

[Fotis Psallidas, Eugene Wu: Smoke: Fine-grained Lineage at Interactive Speed. **PVLDB 2018**]

Why-Provenance

[Boris Glavic: CS595 Data Provenance – Provenance Models and Systems, **Illinois Institute of Technology, 2012**]

- **Overview Why**
  - **Goal:** Which input tuples contributed to an output tuple t in query Q
  - Representation: Set of witnesses w for tuple t (**set semantics!**)
    - $w \subseteq I$ (subset of all tuples in instance I)
    - $t \in Q(w)$ (tuple in result of query over w)

- **Example Witnesses**

```
SELECT Customer, Product
  FROM Orders O, Products P
  WHERE O.PID=P.PID
```

	Customer	Date	PID
o1	A	2019-06-22	2
o2	B	2019-06-22	3
o3	A	2019-06-22	2

	PID	Product
p1	1	X
p2	2	Y
p3	4	Z
p4	3	W

	Customer	Product
t1	A	Y
t2	B	W

  - **Witnesses** for **t1**:
    w1 = {o1,p2}, w2 = {o3,p2},
    w3 = {o1,p2,p3}, …, wn = I
  - Minimal witnesses for t1:
    w1 = {o1,p2}, w2 = {o3,p2}

Others: Where/How Provenance

How-Provenance

[Boris Glavic: CS595 Data Provenance – Provenance Models and Systems, **Illinois Institute of Technology, 2012**]

- **Overview**
  - Model how tuples where combined in the computation
  - **Alternative use:** need one of the tuples (e.g., union/projection)
  - **Conjunctive use:** need all tuples together (e.g., join)

- **Provenance Polynomials**
  - **Semiring annotations** to model provenance $(\mathbb{N}[I], +, \times, 0, 1)$

- **Examples**
  - $q = \pi_a(R)$

	a	b
r1	1	2
r2	1	3

→

a
1

r1 + r2

  - $q = \pi_b(R \bowtie S)$

	a	b
r1	1	P
r2	2	G
	3	M

	c	a
s1	S	1
s2	S	2
s3	W	2

→

b
P
G

Provenance Polynomials

r1 x s1

(r2 x s2) + (r2 x s3)

Why Not?-Provenance

[Adriane Chapman, H. V. Jagadish: **Why not? SIGMOD 2009**]

- **Overview**
  - Why are items not in the results
  - **Example Problem:**
    "Window-display-books < $20"
    → (Euripides, Medea).
    → **Why not** (Hrotsvit, Basilius)?

Author	Title	Price	Publisher
	Epic of Gilgamesh	$150	Hesperus
Euripides	Medea	$16	Free Press
Homer	Iliad	$18	Penguin
Homer	Odyssey	$49	Vintage
Hrotsvit	Basilius	$20	Harper
Longfellow	Wreck of the Hesperus	$89	Penguin
Shakespeare	Coriolanus	$70	Penguin
Sophocles	Antigone	$48	Free Press
Virgil	Aeneid	$92	Vintage

>= 20$?

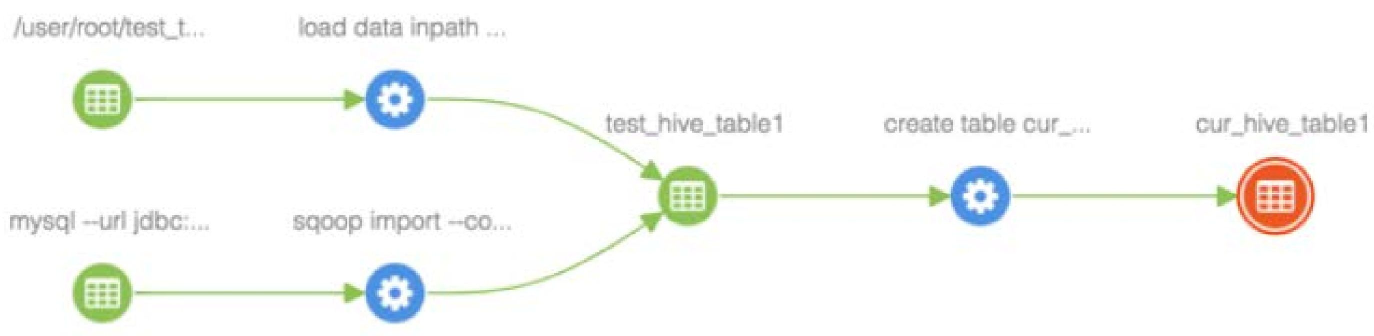
Bug in the query / system?

Not in book store?

- **Evaluation Strategies**
  - Given a user question (why no tuple satisfies predicate S), dataset D, result set R, and query Q, leverage **why lineage**
  - **#1 Bottom-Up:** from leafs in topological order to find last op eliminating $d \in S$
  - **#2 Top-Down:** from result top down to find last op, **requires stored lineage**

Apache Atlas

- **Apache Atlas Overview**
  - Metadata management and governance capabilities
  - Build catalog (data classification, cross-component lineage)
- **Example**
  - Configure Atlas hooks w/ Hadoop components
  - Automatic tracking of lineage and side effects



[https://www.cloudera.com/tutorials/cross-component-lineage-with-apache-atlas-across-apache-sqoop-hive-kafka-storm/.html]

Provenance for ML Pipelines (fine-grained)

- **DEX: Dataset Versioning**
  - **Versioning of datasets, stored with delta encoding**
  - Checkout, intersection, union queries over deltas
  - Query optimization for finding efficient plans

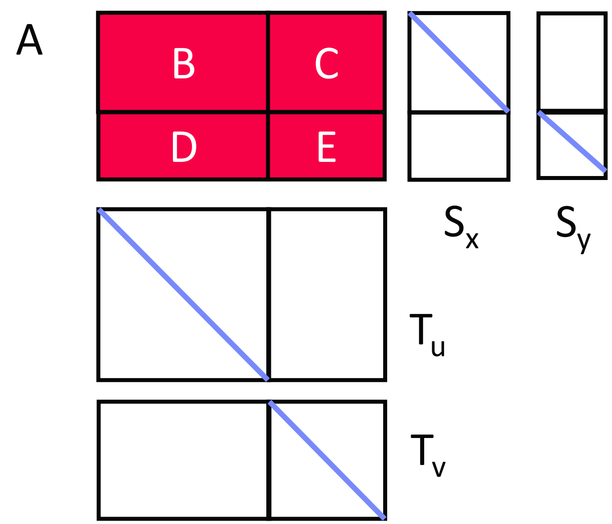
[Amit Chavan, Amol Deshpande: DEX: Query Execution in a Delta-based Storage System. **SIGMOD 2017**]

- **MISTIQUE: Intermediates of ML Pipelines**
  - Capturing, storage, querying of intermediates
  - **Lossy deduplication and compression**
  - Adaptive querying/materialization for finding efficient plans

[Manasi Vartak et al: MISTIQUE: A System to Store and Query Model Intermediates for Model Diagnosis. **SIGMOD 2018**]

- **Linear Algebra Provenance**
  - Provenance propagation by decomposition
  - Annotate parts w/ provenance polynomials (identifiers of contributing inputs + impact)

$$A = S_x B T_u + S_x C T_v + S_y D T_u + S_y E T_v$$



[Zhepeng Yan, Val Tannen, Zachary G. Ives: Fine-grained Provenance for Linear Algebra Operators. **TaPP 2016**]

Provenance for ML Pipelines (coarse-grained)

- **MLflow**
  - Programmatic API for tracking parameters, experiments, and results
  - **autolog()** for specific params

[**Credit:** https://databricks.com/blog/2018/06/05]

```python
import mlflow
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)
mlflow.log_metric("accuracy", 0.1)
mlflow.log_artifact("roc.png")
```

- **Flor (on Ground)**
  - DSL embedded in python for managing the workflow development phase of the ML lifecycle
  - DAGs of Actions, Artifacts, and Literals
  - Data context generated by activities in Ground

[Credit: https://rise.cs.berkeley.edu/projects/jarvis/]

[Joseph M. Hellerstein et al: Ground: A Data Context Service. **CIDR 2017**]

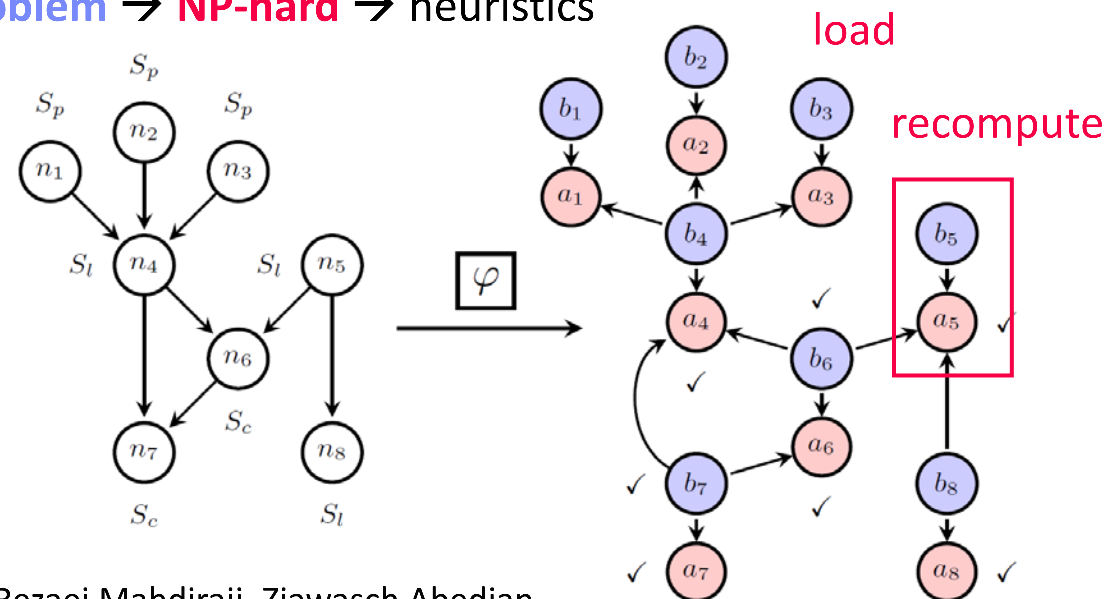
- **Dataset Relationship Management**
  - **Reuse**, **reveal**, **revise**, **retarget**, **reward**
  - Code-to-data relationships (data provenance)
  - Data-to-code relationships (potential transforms)

[Zachary G. Ives, Yi Zhang, Soonbo Han, Nan Zheng,: Dataset Relationship Management. **CIDR 2019**]

Provenance for ML Pipelines (coarse-grained), cont.

- **HELIX**
  - Goal: focus on iterative development w/ small modifications (trial & error)
  - Caching, reuse, and recomputation
  - Reuse as **Max-Flow problem** → **NP-hard** → heuristics
  - Materialization to disk for future reuse

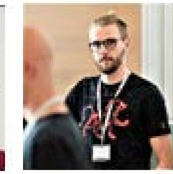
[Doris Xin, Stephen Macke, Litian Ma, Jialin Liu, Shuchen Song, Aditya G. Parameswaran: Helix: Holistic Optimization for Accelerating Iterative Machine Learning. **PVLDB 2018**]



- **Collaborative Optimizer**

[Behrouz Derakhshan, Alireza Rezaei Mahdiraji, Ziawasch Abedjan, Tilmann Rabl, Volker Markl: Optimizing Machine Learning Workloads in Collaborative Environments. **SIGMOD 2020**]

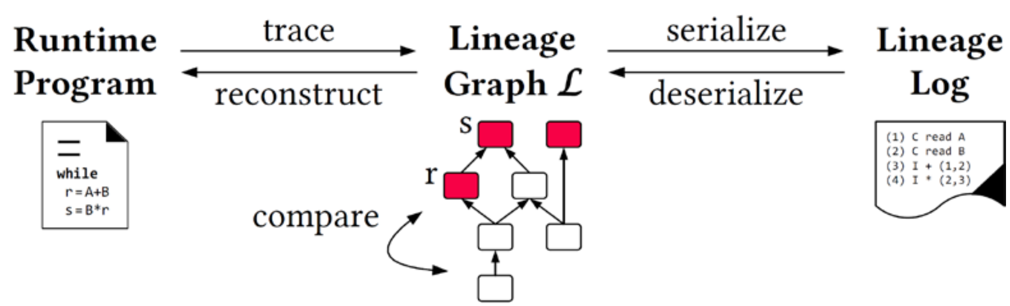
Lineage Tracing & Reuse in SystemDS

- **Problem**
  - **Exploratory data science** (data preprocessing, model configurations)
  - **Reproducibility** and **explainability** of trained models (data, parameters, prep)

  ➔ **Lineage/Provenance as Key Enabling Technique:**
  Model versioning, reuse of intermediates, incremental maintenance, auto differentiation, and debugging (query processing over lineage)

- **Efficient Lineage Tracing**
  - Tracing of inputs, literals, and **non-determinism**
  - **Trace lineage of logical operations**
  - **Deduplication** for loops/functions
  - Program/output reconstruction



[Arnab Phani, Benjamin Rath, Matthias Boehm: LIMA: Fine-grained Lineage Tracing and Reuse in Machine Learning Systems, **SIGMOD 2021**]

Lineage Tracing & Reuse in SystemDS

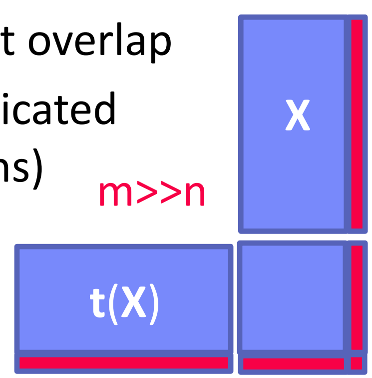
- **Multi-level, Lineage-based Reuse**
  - Lineage trace uniquely identifies intermediates
  - Reuse intermediates at function / block / operation level

- **Full Reuse of Intermediates**
  - Before executing instruction, probe output lineage in cache Map<Lineage, MatrixBlock>
  - Cost-based/heuristic caching and eviction decisions (compiler-assisted)

```
for( i in 1:numModels )
  R[,i] = lm(X, y, lambda[i,], ...)
```

```
m_lmDS = function(...) {
  l = matrix(reg,ncol(X),1)
  A = t(X) %*% X + diag(l)
  b = t(X) %*% y
  beta = solve(A, b) ...}
```

- **Partial Reuse of Intermediates**
  - **Problem:** Often partial result overlap
  - Reuse partial results via dedicated rewrites (compensation plans)
  - Example: steplm




```
m_steplm = function(...) {
  while( continue ) {
    parfor( i in 1:n ) {
      if( !fixed[1,i] ) {
        Xi = cbind(Xg, X[,i])
        B[,i] = lm(Xi, y, ...)
    } }
    # add best to Xg
    # (AIC)
} }
```

BREAK (and Test Yourself)

- **Given below tables R and S (with tuples r_i and s_i), query Q and results O, specify the provenance polynomials for every tuple in O.** (3 points)

R

	A	B
r_1	X	1
r_2	Y	2
r_3	Z	1

S

	C	D
s_1	1	A
s_2	2	B
s_3	2	A
s_4	2	C

```
SELECT DISTINCT S.D
  FROM R, S
  WHERE R.B=S.C
```

O	Provenance Polynomials?
A	
B	
C	

A: r1 x s1 + r3 x s1 + r2 x s3
(equivalent: **(r1 + r3) x s1 + r2 x s3**)

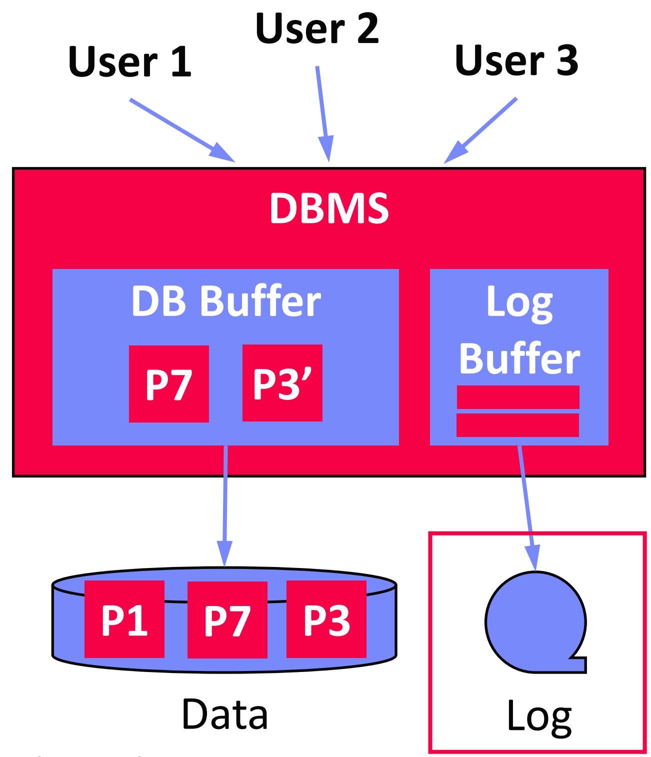
B: r2 x s2

C: r2 x s4

Blockchain Fundamentals

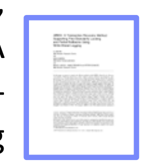
Recap: Database (Transaction) Log

- **Database Architecture**
  - **Page-oriented storage** on disk and in memory (DB buffer)
  - Dedicated **eviction algorithms**
  - Modified in-memory pages marked as dirty, flushed by cleaner thread
  - **Log:** append-only TX changes
  - Data/log often placed on different devices and periodically archived (backup + truncate)



- **Write-Ahead Logging (WAL)**
  - The log records representing changes to some (dirty) data page must be on **stable storage before the data page** (UNDO - atomicity)
  - **Force-log on commit** or full buffer (REDO - durability)
  - **Recovery:** forward (REDO) and backward (UNDO) processing
  - Log sequence number (LSN)

[C. Mohan, Donald J. Haderle, Bruce G. Lindsay, Hamid Pirahesh, Peter M. Schwarz: ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging. **TODS 1992**]



Bitcoin and Blockchain

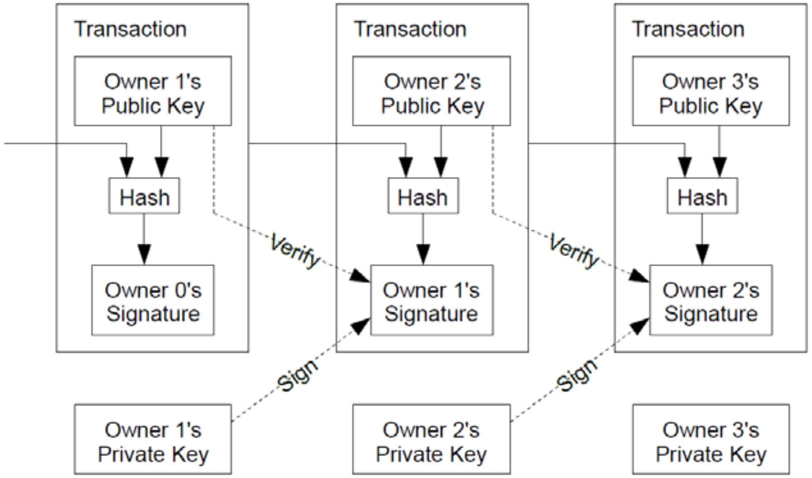
[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Motivation**
  - Peer-to-peer (decentralized, anonymous) electronic cash/payments
  - **Non-reversible transactions** w/o need for trusted third party
  - **Statistics**

	Market Price (USD)	Average Block Size	Transactions per Day	Mempool Size
Nov 21 2019:	$7,862.72 USD	1.16 Megabytes	303,921 Transactions	11,304,890 Bytes
Nov 19 2020: (Paypal Oct 2020)	$17,975.24 USD	1.29 Megabytes	310,424 Transactions	19,920,773 Bytes

[https://www.blockchain.com/charts]

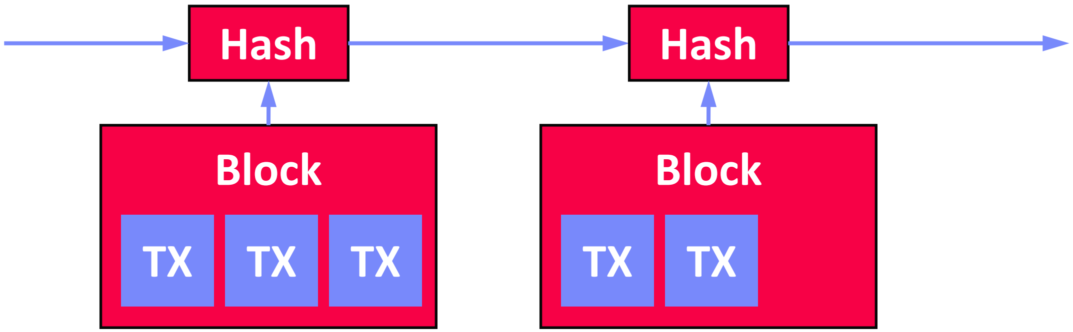
- **Transaction Overview**
  - Electronic coin defined as chain of digital signatures
  - Transfer by signing hash of previous TX and public key of next owner
  - **Double-spending problem** (without global verification)



Blockchain Data Structure

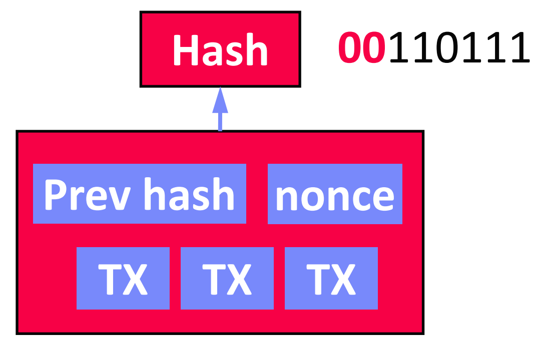
[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Timestamp Server**
  - Decentralized timestamp server: chain of hashes ➔ **public ledger**



Enforces order dependency
➔ No double-spending

- **Proof-of-Work**
  - Scanning for value (nonce) which **SHA-256 hash** begins with a number of zero bits
    → exponential in number of zeros
  - # zero bits determined by MA of avg blocks/hour
  - Hard to recompute for chain, easy to check
  - **Majority decision:** CPU time, longest chain



Merkel tree (hash tree)

Blockchain Data Structure, cont.

- **Bitcoin Mining**
  - HW: from CPU to GPUs/FPGAs/ASICs (**10-70 TH/s** @ 2-3KW)
  - Usually mining pools → **"mining cartels"**

@Malaysia

- **Hash Rate of Bitcoin Network**
  - **~10 min per block** (144 blocks per day)

Nov 12, 2021: ~160EH

[https://www.blockchain.com/en/charts/hash-rate?daysAverageString=7&timespan=180days, **Nov 12 2021**]

Blockchain Communication

[**Satoshi Nakamoto**: Bitcoin: A Peer-to-Peer Electronic Cash System, **White paper 2008**]

- **Networking Protocol**
  - New **TXs are broadcast** to all nodes
  - Each node collects new **TXs into a block**
  - Each node works on finding **proof-of-work** for its block
    → **Incentive:** 1st TX in block new coin (halves every 210k blocks) for the block creator + TX fees
  - When a node finds a proof-of-work, **broadcast the block** to all nodes
  - Nodes accept the block if **all TXs are valid** (double spending)
  - Nodes express **acceptance by working on next block** in the chain, using the hash of the accepted block as the previous hash

2008: 50BTC
2012: 25BTC
2016: 12.5BTC
2020: 6.25BTC

- **Fault Tolerance**
  - **TX broadcasts:** no need to reach all but many → next block contains it
  - **Block broadcast:** no need to reach all → next block references it

Smart Contracts (Ethereum)

- **Motivation**
  - **Problem:** Bitcoin TXs for transferring X $BTC from A to B (exchange as assets)
  - **Goals:** voting, auctions, games, bets, legal agreements (notary)
- **Ethereum**
  - Decentralized platform that allows creation, management, and execution of smart contracts
  - Ether cryptocurrency, block mining rate: seconds (5 ETH/block)

[**Credit:** Shana Hutchison]

- **Smart Contract**
  - Store smart contract (turing-complete programs) on the blockchain
  - **On transfer/trigger:** run smart contract (in) in Ethereum Virtual Machine
  - Language: Serpent/Solidity (deterministic, w/ control flow and function calls) ➔ Problem: while(true) → EVM gas and fees (start gas, gas price)
  - Immutability guarantees persistence

Consensus Mechanisms

"means of showing that one invested a non-trivial amount of effort related to some statement"

- **Proof of Work (PoW)**
  - Validation by performing work, existence of HW resources
  - **High HW cost of attacks**
  - **Wasted work, resources, energy** (only first block, no real outcome, e-waste)
- **Proof of Stake (PoS)**
  - Validation by stake-weighted random node selection
  - **Intrinsic coin cost**, **less HW resources**/energy
  - **Untested attack mitigation?**

Ethereum 2.0
→ PoS/sharding over time

- **Proof of Space/Capacity**
  - Upfront creation of "plot files", store nonces+hashes, find solutions, occasional validation
  - **HW costs of attacks**, use of **unused space**
  - **Moderate adoption**

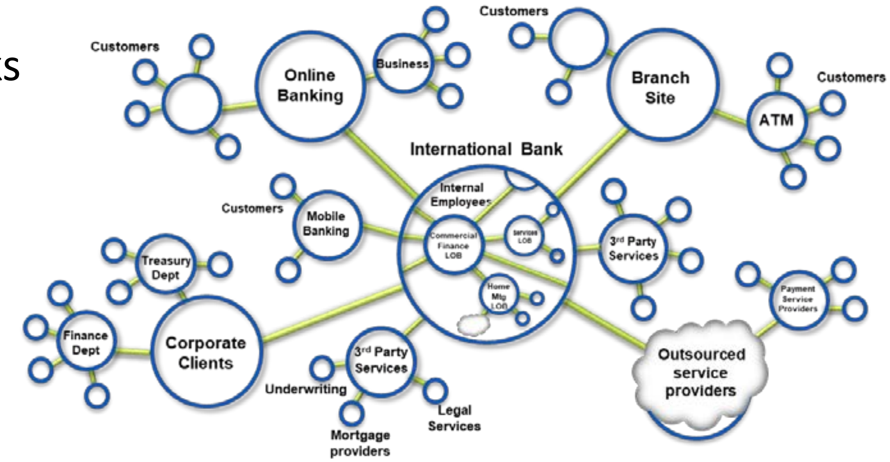
[https://www.chia.net/]

[Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, Krzysztof Pietrzak: Proofs of Space. IACR Cryptol. 2013]

Permissioned/Private Blockchains

[**C. Mohan:** State of Permissionless and Permissioned Blockchains: Myths and Reality, **2019**]

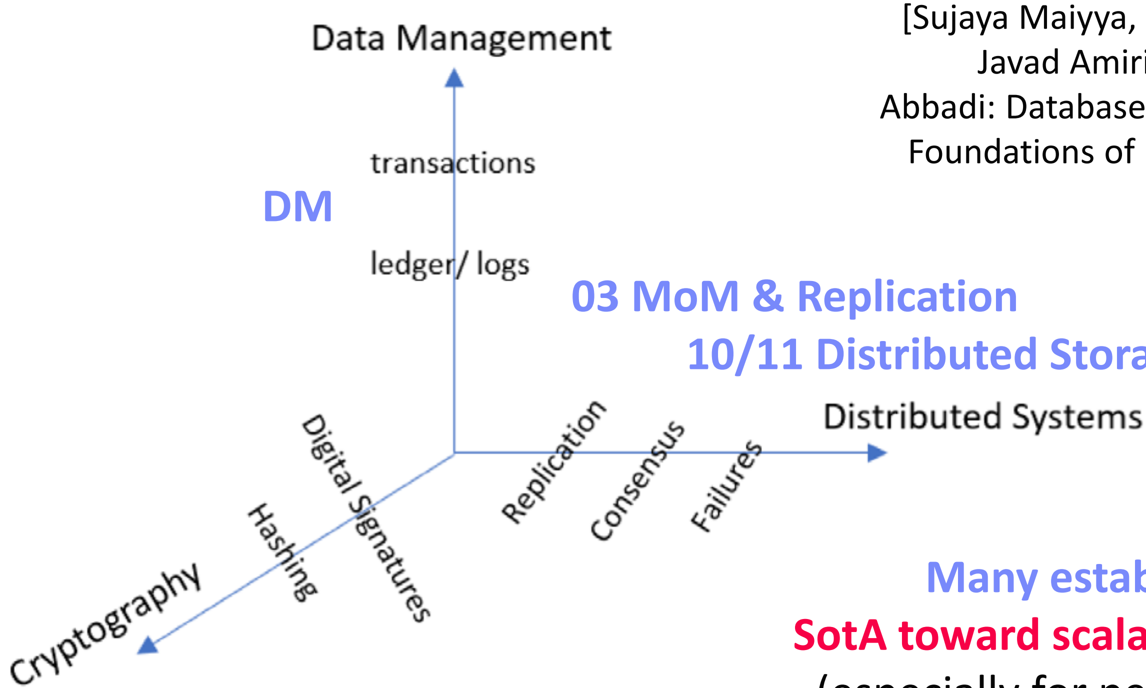
- **Private Setup**
  - Business Networks connect businesses
  - **Participants with Identity**
  - Assets flow over business networks
  - Transactions describe exchange or change of states of assets
  - **Smart contracts** underpin transactions
  - Blockchain as shared, replicated, permissioned ledger (TX log): **consensus**, **provenance**, **immutability**



- **Hyperledger Fabric (https://github.com/hyperledger/)** HYPERLEDGER
  - IBM, Oracle, Baidu, Amazon, Alibaba, Microsoft, JD, SAP, Huawei, Tencent
  - **Blockchain-as-a-Service** (BaaS) offerings: distributed ledger, libs, tools

Discussion Blockchain

[Sujaya Maiyya, Victor Zakhary, Mohammad Javad Amiri, Divyakant Agrawal, Amr El Abbadi: Database and Distributed Computing Foundations of Blockchains. **SIGMOD 2019**]



03 MoM & Replication

10/11 Distributed Storage/Compute

Many established techniques

SotA toward scalable/efficient blockchains

(especially for permissioned blockchains)

➔ **Recommendation: Investigate business requirements/context, decide on technical properties and acceptable trade-offs**

Summary and Q&A

- **Summary**
  - Motivation and Terminology
  - Data Provenance
  - Blockchain Fundamentals

- **Next Lectures (Part B)**
  - **08 Cloud Computing Foundations** [Nov 27]
  - **09 Cloud Resource Management and Scheduling** [Dec 04]
  - **10 Distributed Data Storage** [Dec 11]