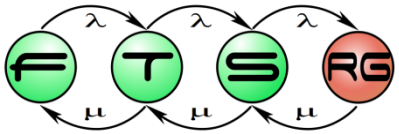
Can we trust blockchain as a public utility? - The performability aspect.

András Pataricza

With contributions of
I. Kocsis, A. Klenik, Sz. Bozoki, A. Foldvari, D. Burjan, L. Gonczy, M. Telek, G. Horvath

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



Blockchain@BME MIT

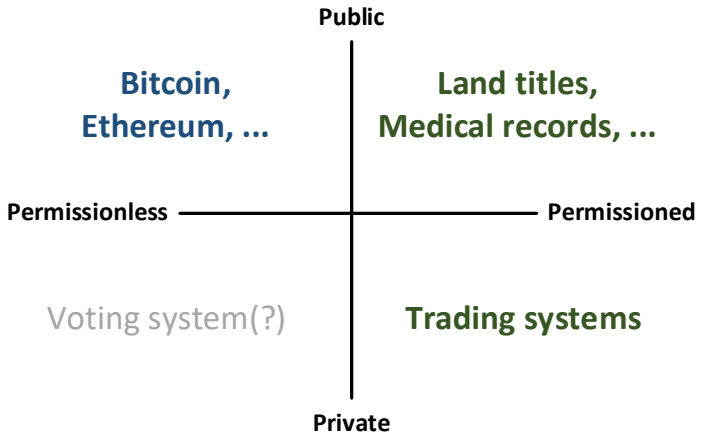
- IBM Faculty Award 2016,
- Summer internship at the Linux Foundation: "*Hyperledger as a Business Process Execution Engine*"
- Linux Foundation + Hyperledger project membership
  - Membership in Performance and Scalability Working Group
- ISO standardization (Hungarian representative in blockchain)
- Course: "Blockchain technologies and applications" – 2018 Spring: 200 students

PERMISSIONED AND/OR NONPUBLIC BLOCKCHAINS

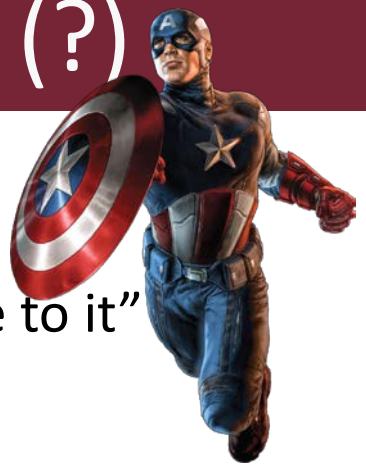
Key differences - informally

	Network	Participants	Consensus	Transactions
(Bit/alt)coin				
Nonpublic/ permissioned blockchain				

Permissioned/-less:
who can write the ledger
Public/private(/consortial):
who can read the ledger



Standards to the rescue! (?)



- **NISTIR 8202** (DRAFT): Blockchain Technology Overview
- No notion of “private/public” (yet?)
- Permissioned: “only particular users can read and write to it”

- Permissioned blockchains (excerpts):
  - defy the original conception of the Bitcoin blockchain
  - Where everyone can read and write […] the ledger is transparent/public

  - **Organizations** that wish to work together, but **do not fully trust** one another
  - invite […] partners to record […] transactions on a **shared distributed ledger**

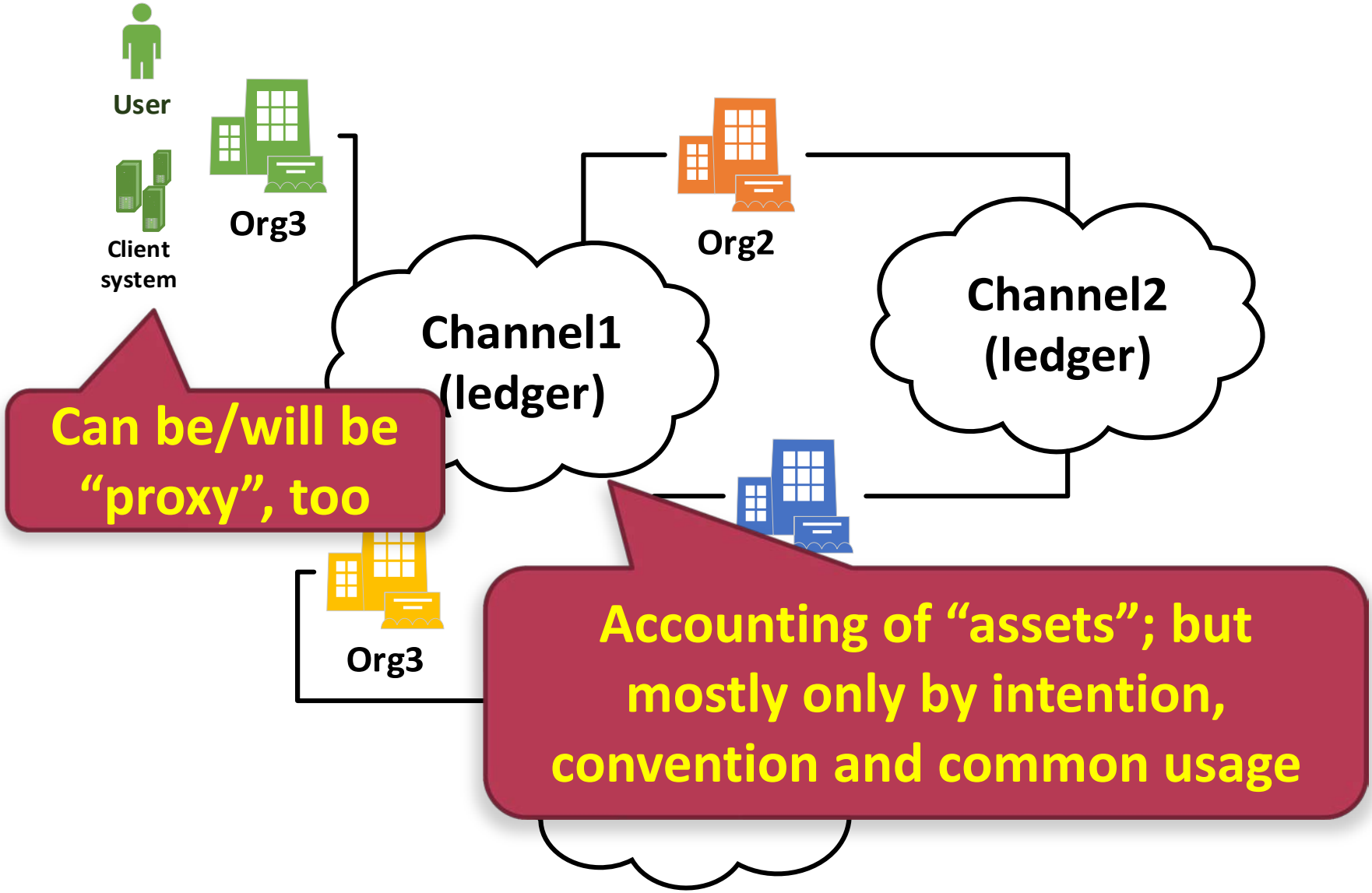
  - **same traceability** of assets as they pass through the blockchain
  - **same** distributed, resilient, and redundant **data storage system**

  - These organizations can determine the **consensus mechanism** to be used,
  - based on how much they **trust** one another

HYPERLEDGER FABRIC

Select slides taken from sources available at:
https://wiki.hyperledger.org/community/presentations

Basic model


User
Client system
Org3
Org2
Channel1 (ledger)
Channel2 (ledger)
Org3
Can be/will be "proxy", too
Accounting of "assets"; but mostly only by intention, convention and common usage

Smart contracts?

- Deployed on channel
- Called ***chaincode***
- Sees ledger as (versi

Recall: the "just a dumb database" interpretation of private/permissioned Blockchains

- No (built-in) cryptocurrency
  - No "native transaction type"
  - The system is meaningless w/o chaincodes

- Remember the token taxonomy?
  - That works here, too – even coins
  - ICO's – maybe in dampened, "privatized" forms
  - You *can* anchor to public chains

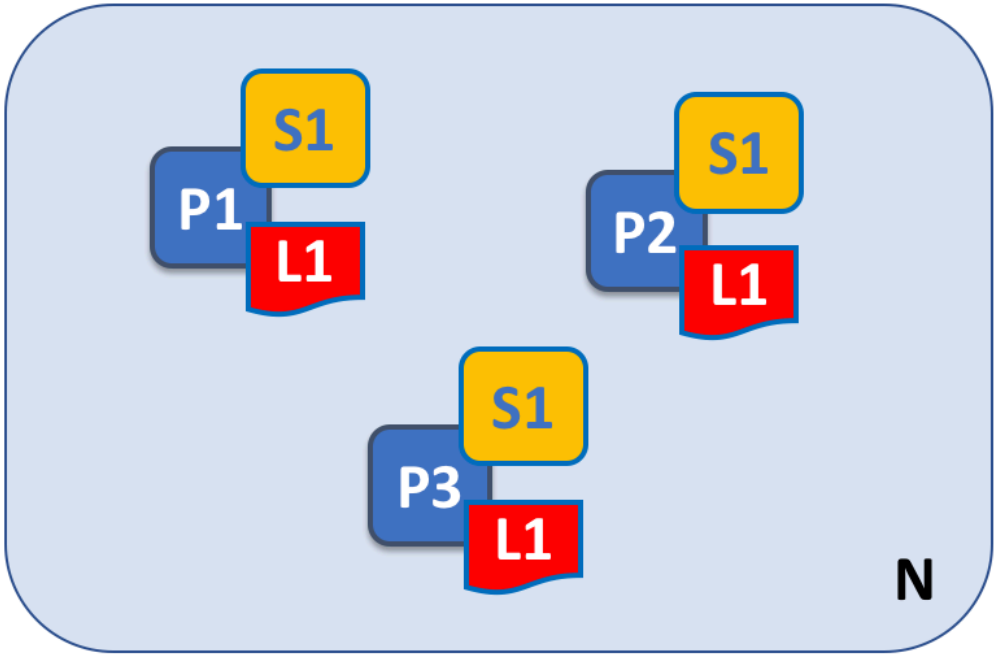
Fabric - smart contract style

```
func (s *SmartContract) changeCarOwner(APIst                    {

        if len(args) != 2 {
                return shim.Error("Incorrec
        }

        carAsBytes, _ := APIstub.GetState(a
        car := Car{}

        json.Unmarshal(carAsBytes, &car)
        car.Owner = args[1]

        carAsBytes, _ = json.Marshal(car)
        APIstub.PutState(args[0], carAsBytes)

        return shim.Success(nil)
}
```

Noteworthy:
"Naked" KV store interface
"Token" emerges from domain model
No addresses, no "payable"
No "gas" – just timeout
Not shown, but present: identities
More straightforward then Solidity
Could be Java/JavaScript/…

Source: https://github.com/hyperledger/fabric-samples/blob/release/chaincode/fabcar/fabcar.go

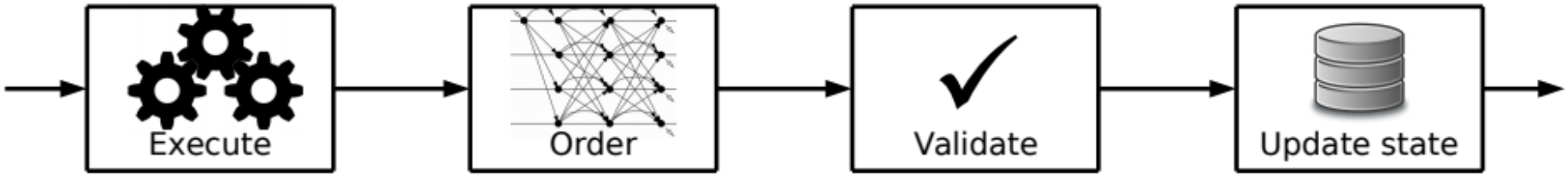
Where are the computers? – Peers, ledgers and chaincodes



N	Blockchain network
P	Peer node
S	Smart contract (aka chaincode)
L	Ledger

Source of the figures: http://hyperledger-fabric.readthedocs.io/en/release-1.1/peers/peers.html

Execute-order-validate architecture



- Simulate trans. and endorse
- Create rw-set
- Collect endorse-ments

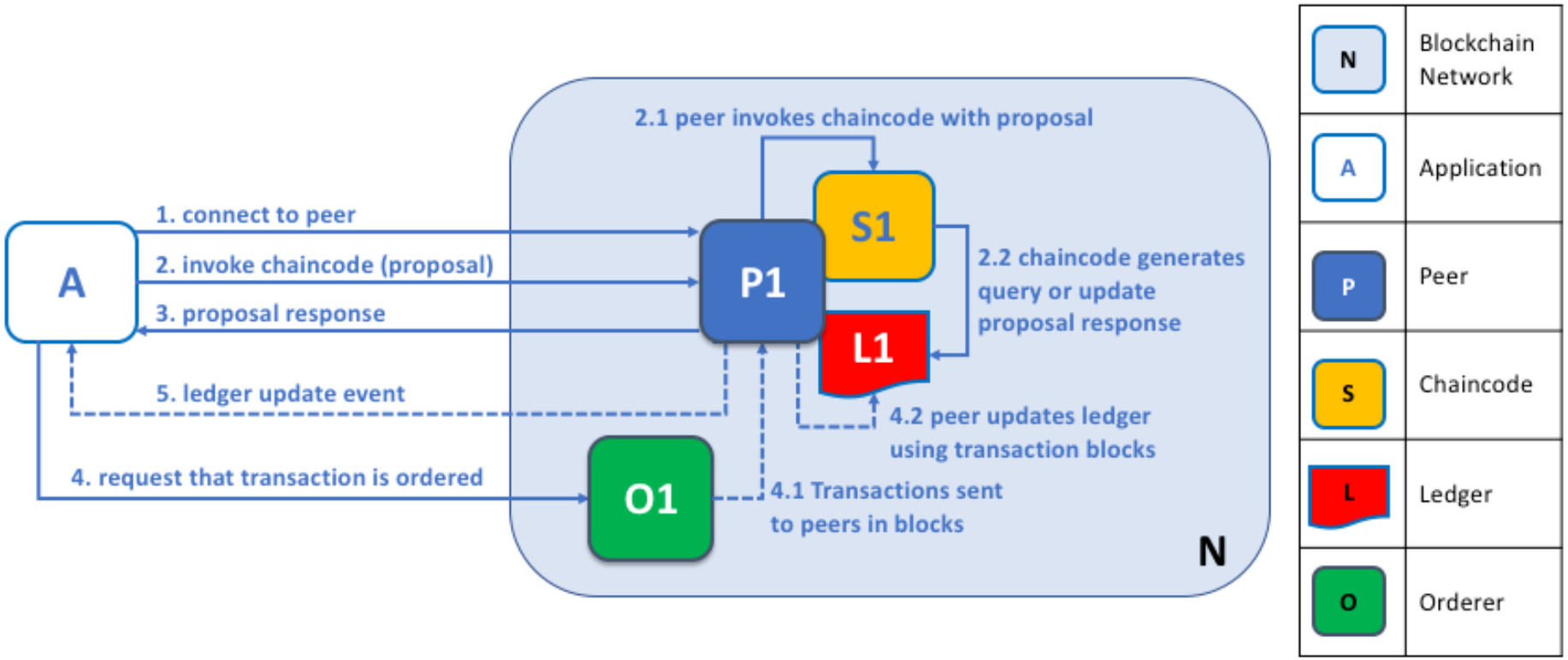
- Order rw-sets
- Atomic broadcast (consensus)
- Stateless ordering service

- Validate endorse-ments & rw-sets
- Eliminate invalid and conflicting trans.

- Persist state on all peers

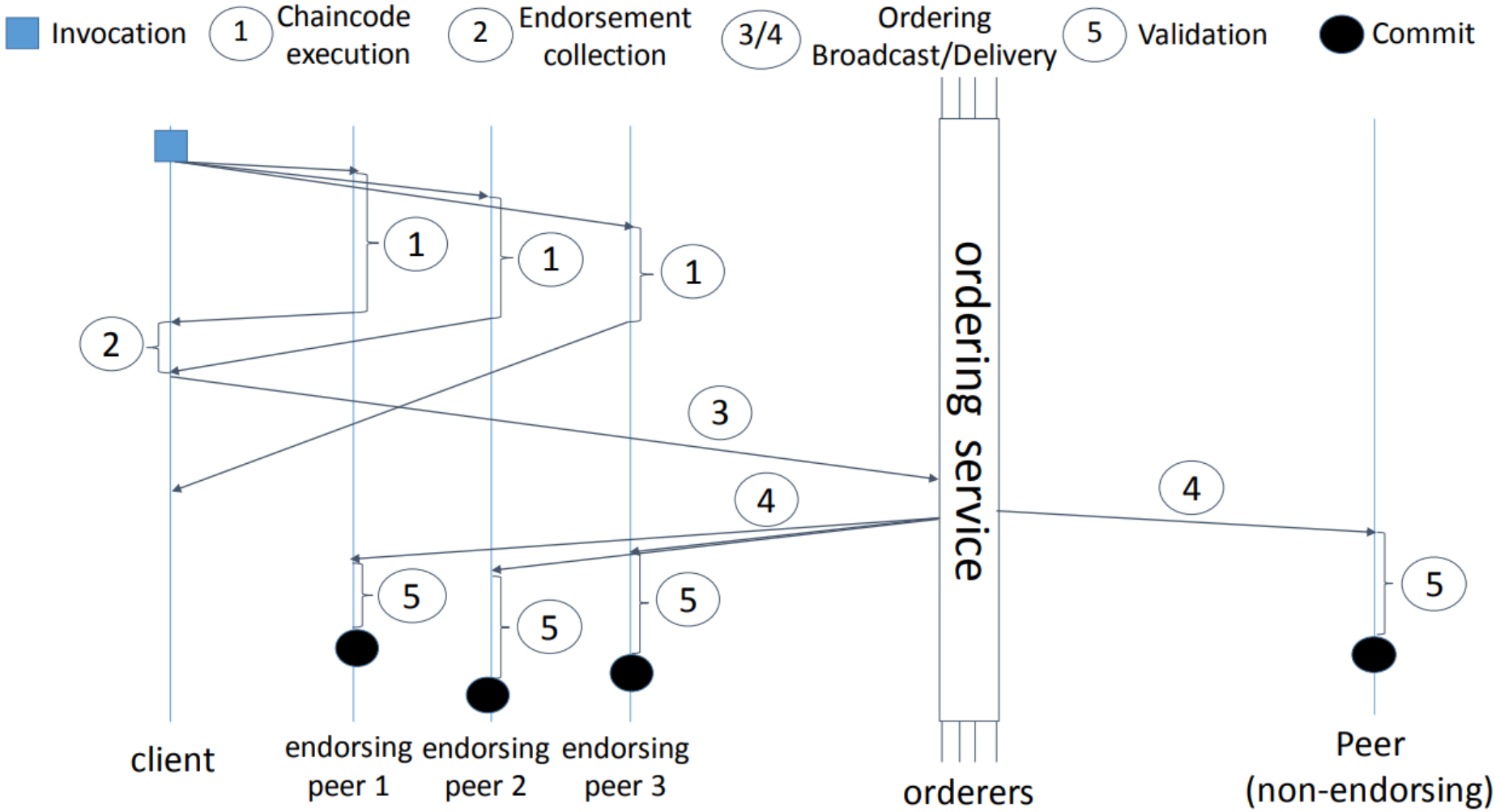
https://arxiv.org/pdf/1801.10228v1.pdf

Proposing a ledger change



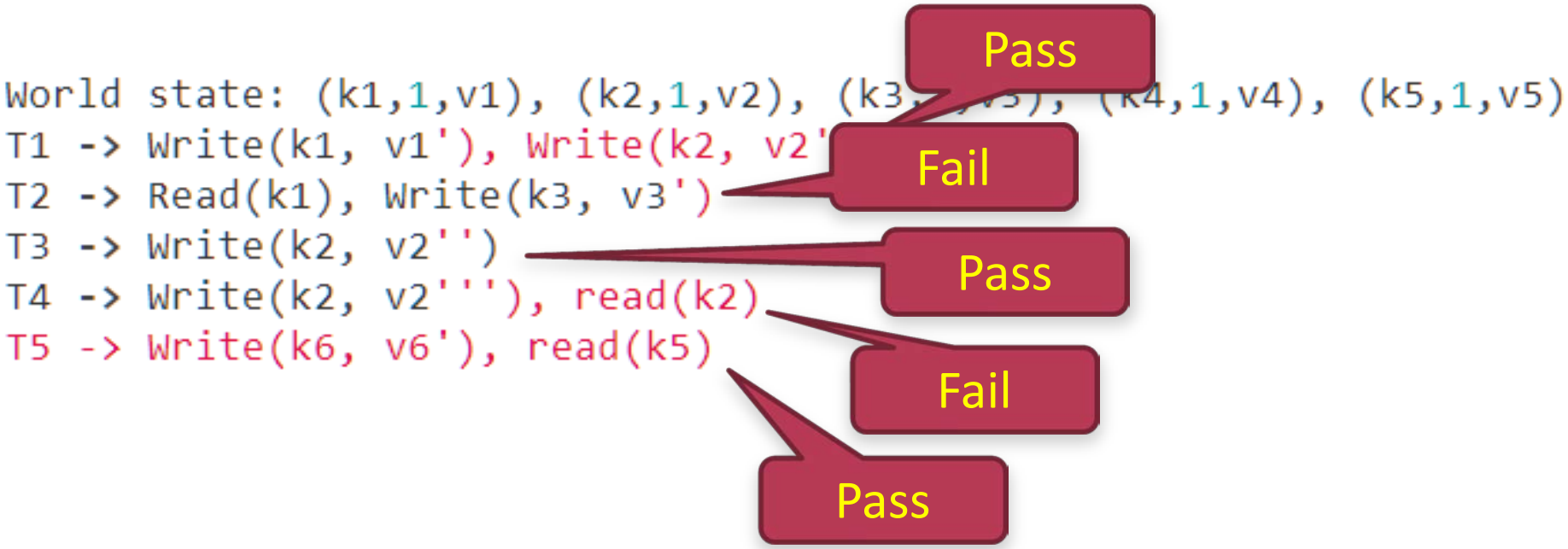
A bit like doing any "official" process in real life

Transaction flow



https://arxiv.org/pdf/1801.10228v1.pdf

Validation based on Tx read-write sets



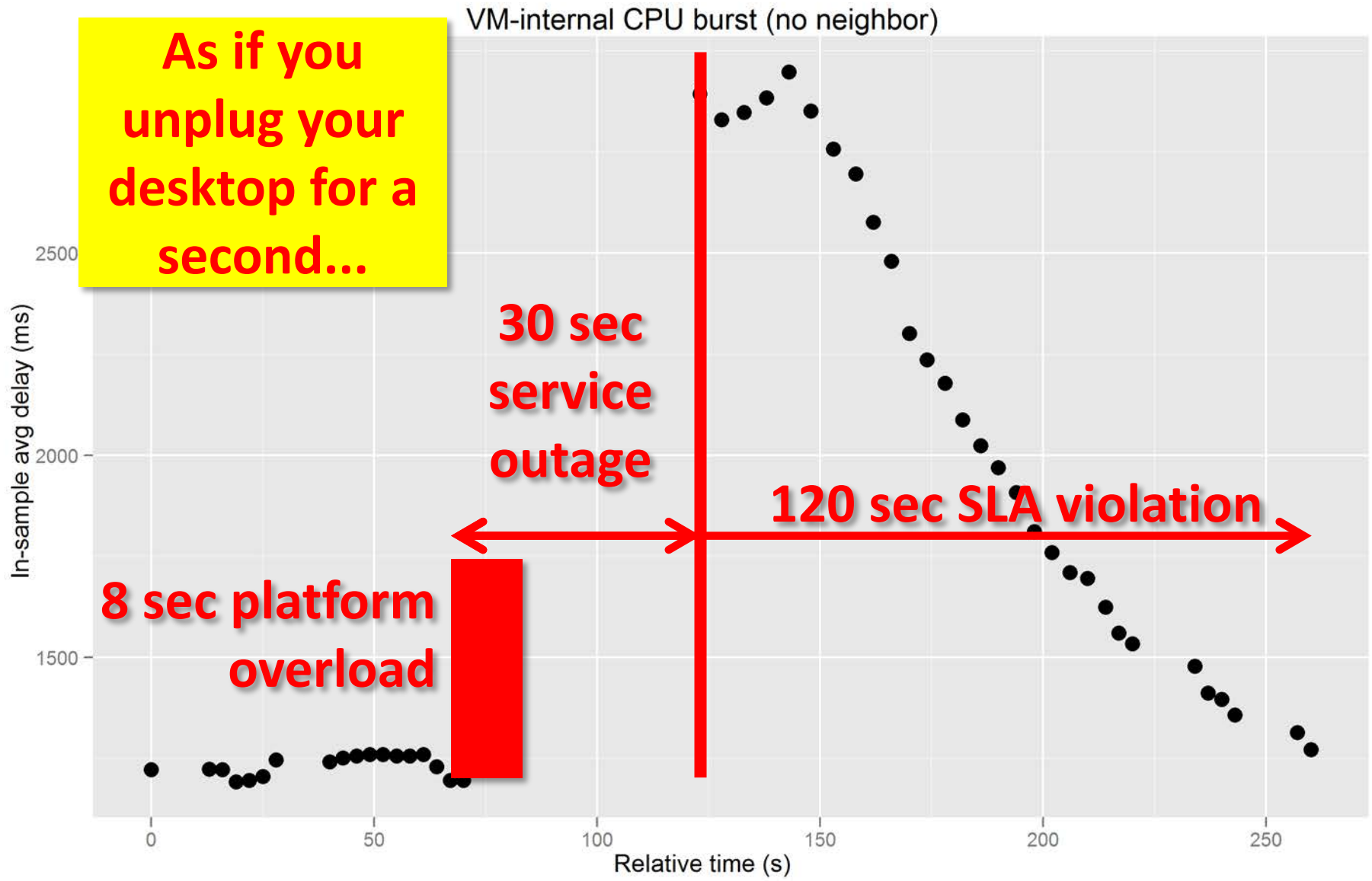
We've seen this before – concurrent reads and writes

Fabric: key properties

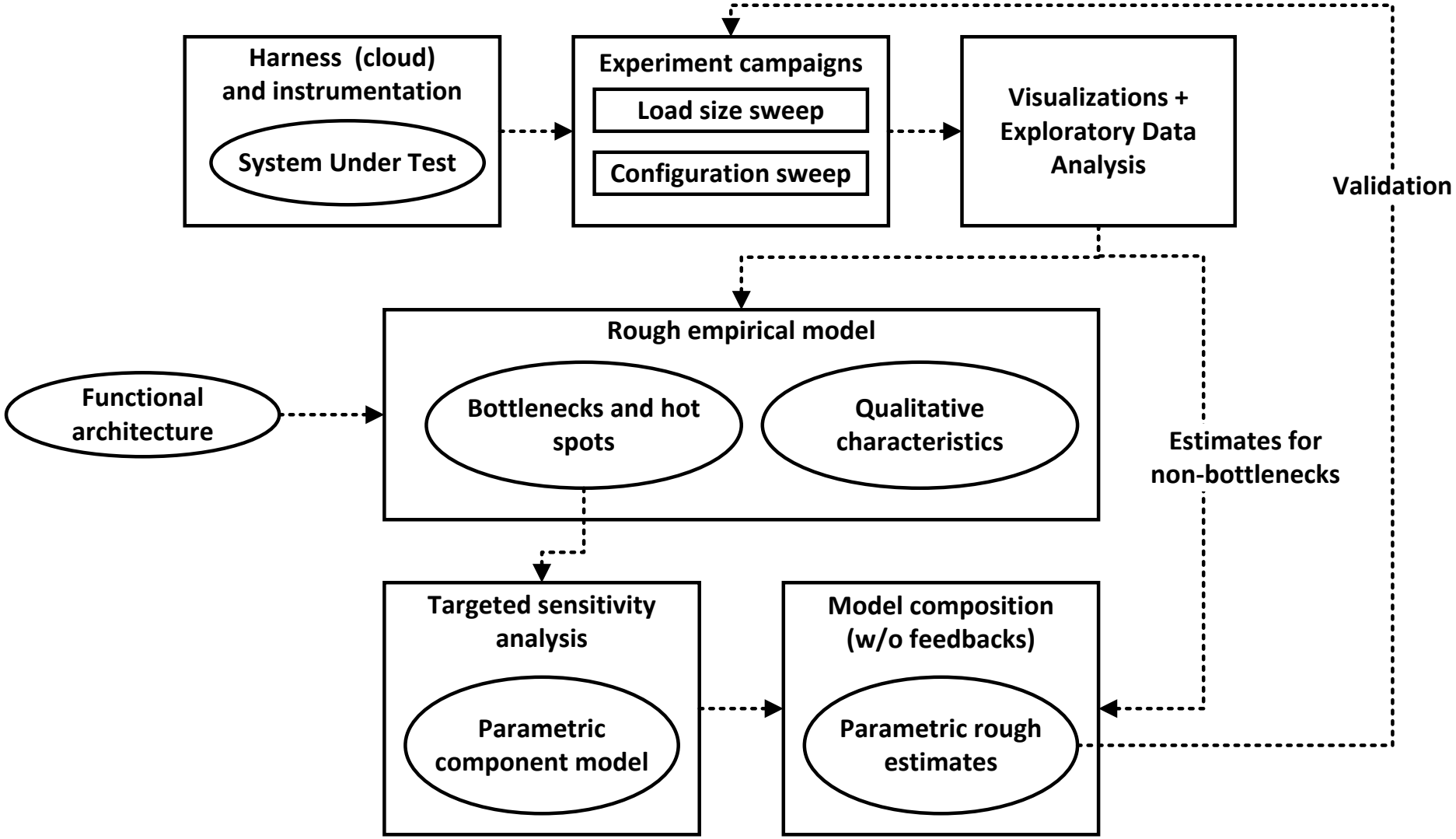
- "Distributed ledger"
  - Maintaining K-V store replicas in a ***permissioned P2P system***
  - Blockchain-backed: ***same log of all transactions /peer*** (Bitcoin-style)
- Chaincode: ***fenced off, ~arbitrary Tx code over the Ledger***
- *Key (inferred) requirements: throughput, scalability, consistency*
- ***Distributed consensus***
- Part of the ***Hyperledger*** umbrella project
- ***Our research: 1.1, upcoming: 1.2***

PERFORMABILITY BENCHMARKING

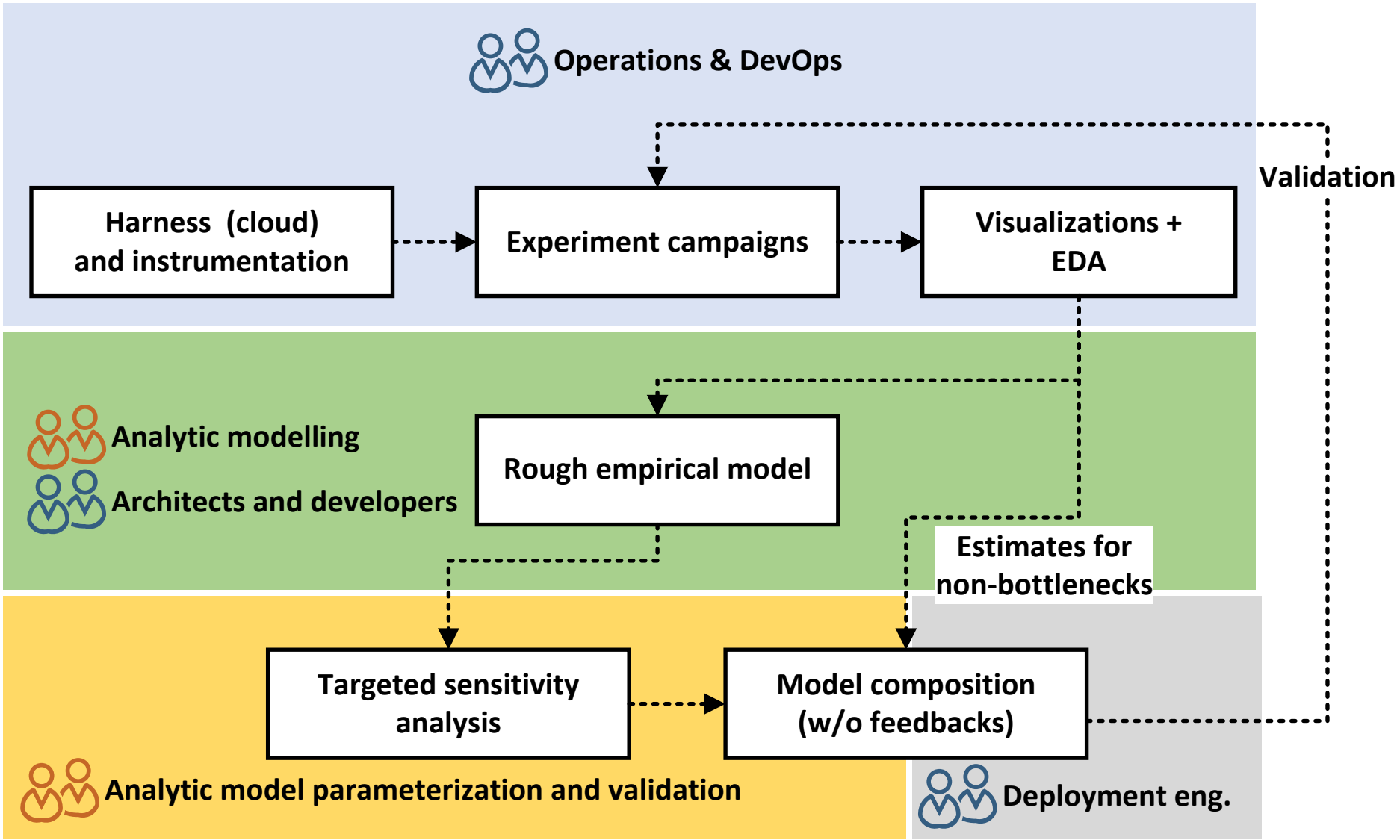
Short transient faults – long recovery



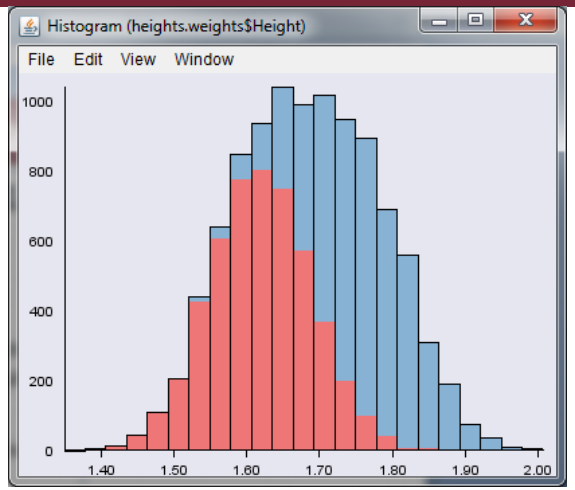
Overview of our approach

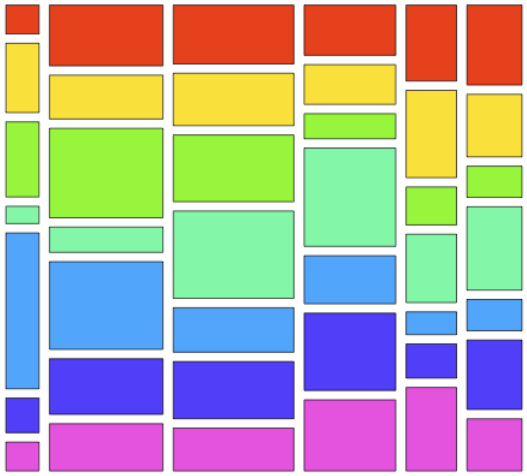


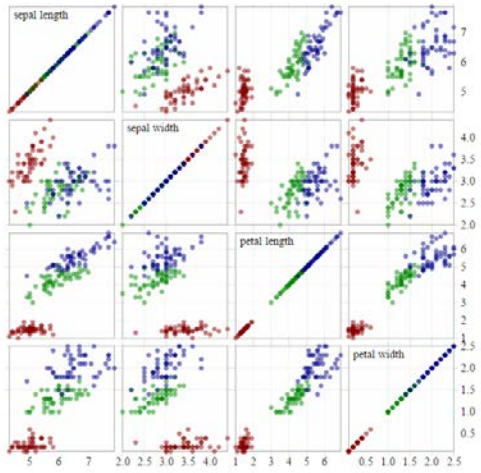
Potential benefits

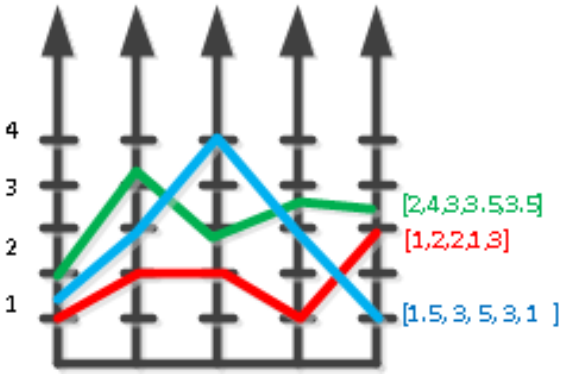


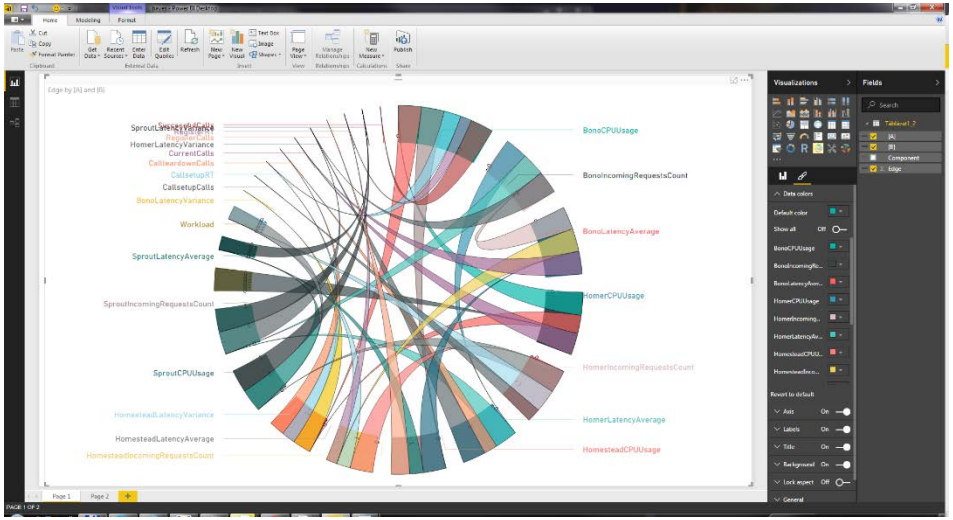
Visual analytics & statistics



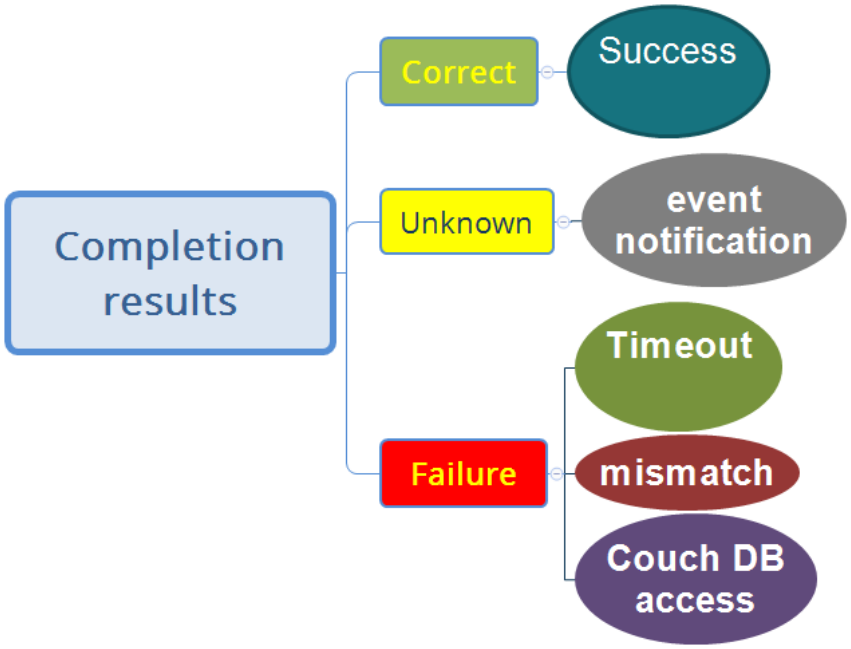


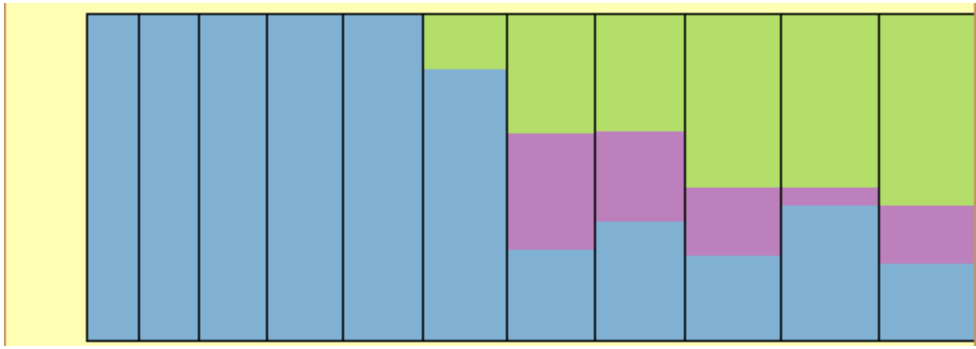


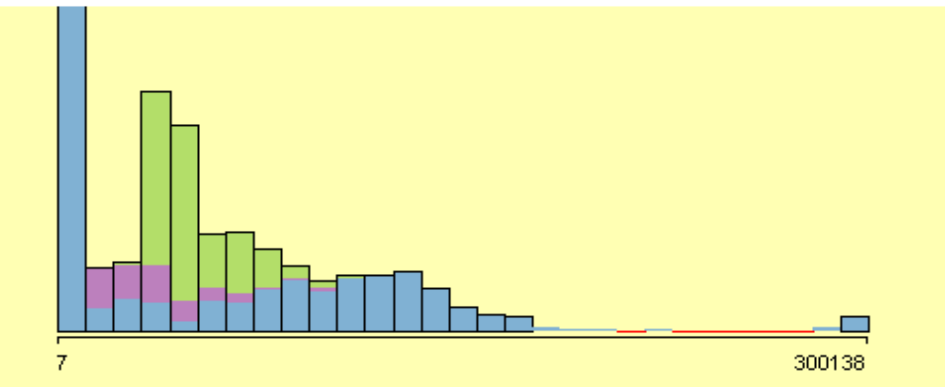


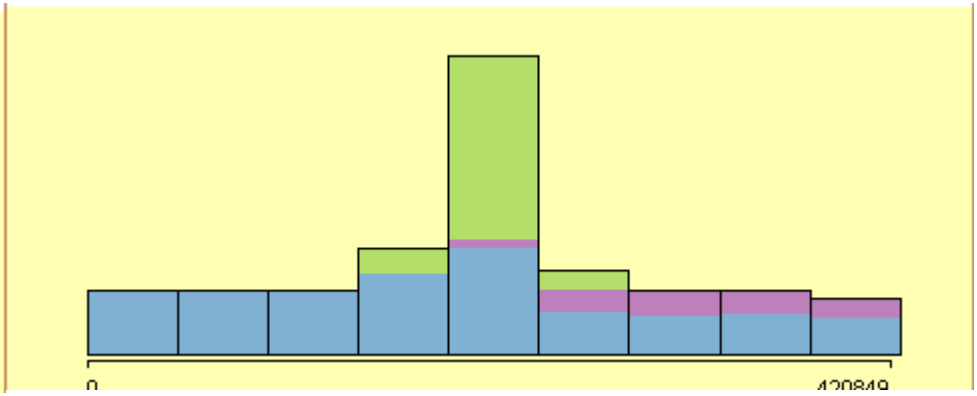


Failure EDA

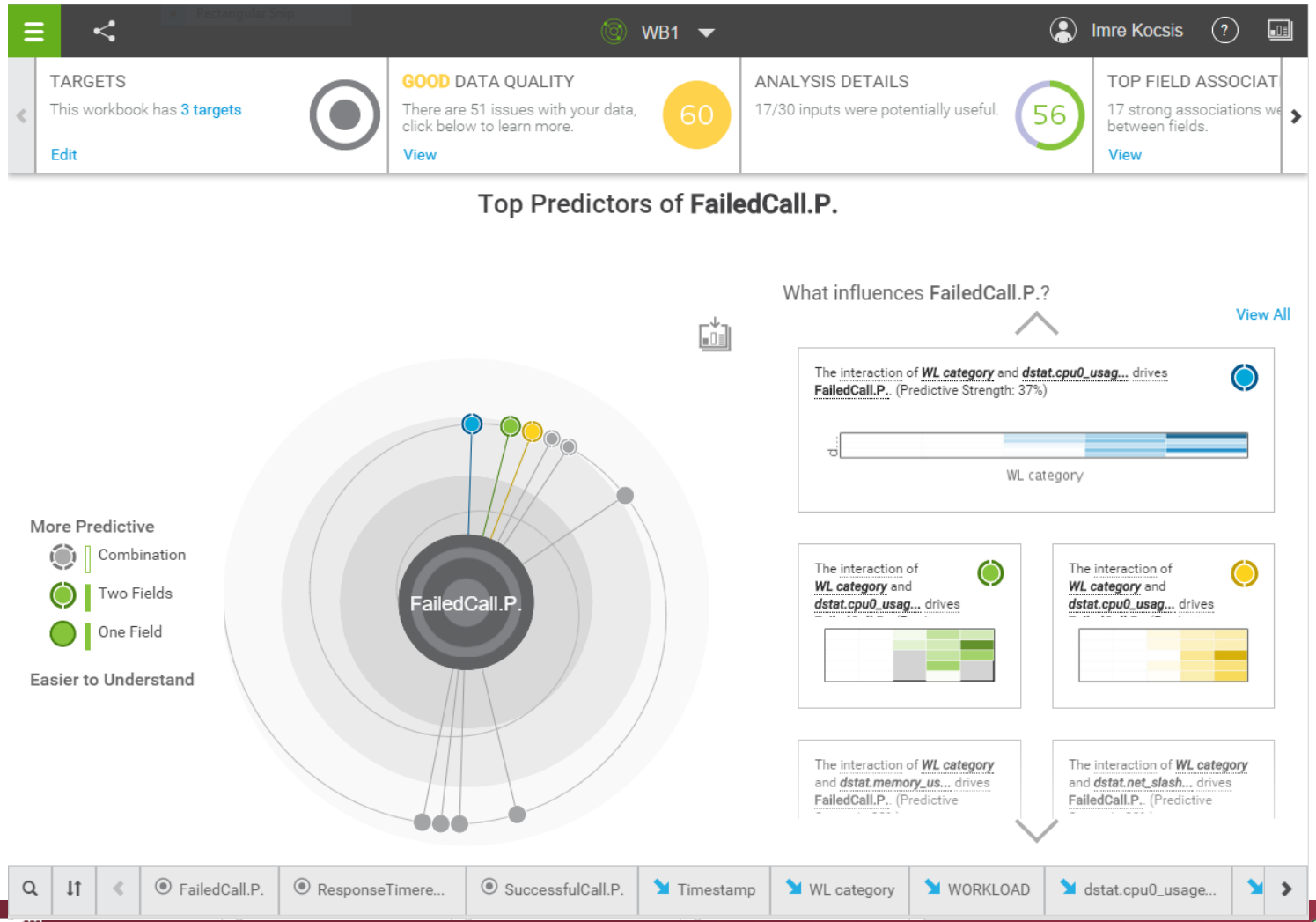

Completion results
Correct
Success
Unknown
event notification
Failure
Timeout
mismatch
Couch DB access




7
300138


0
420849

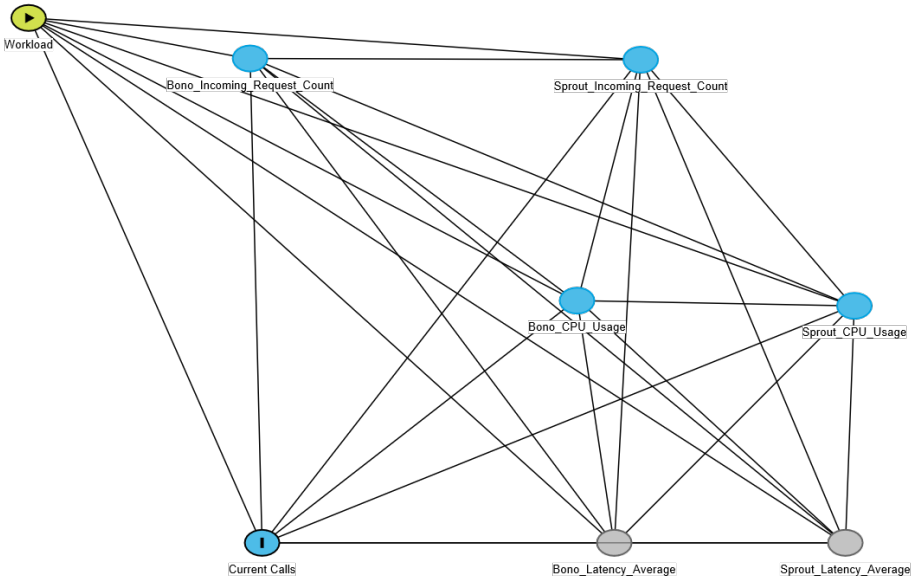
Business intelligence for automated discovery of qualitative hypotheses



Causal Bayesian Analysis

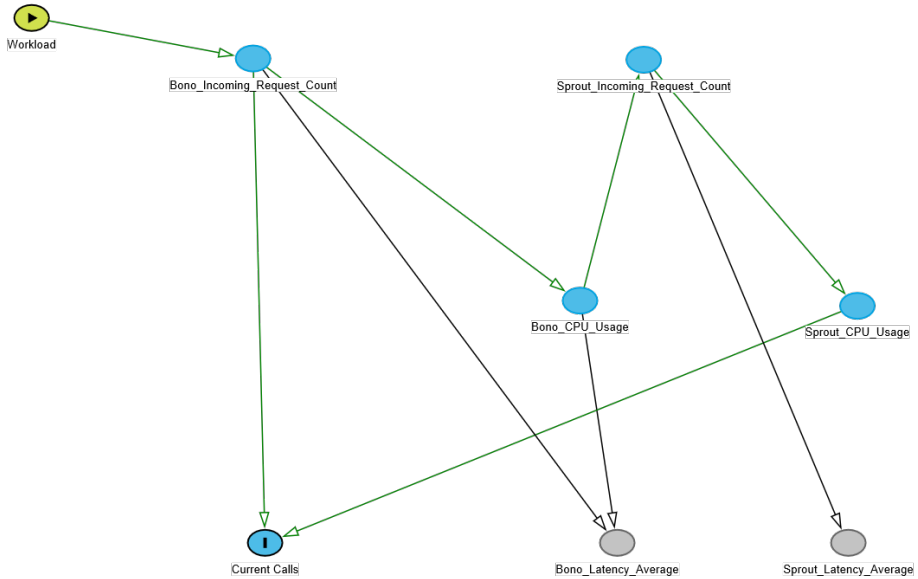
Correlation graph

(phenomena)



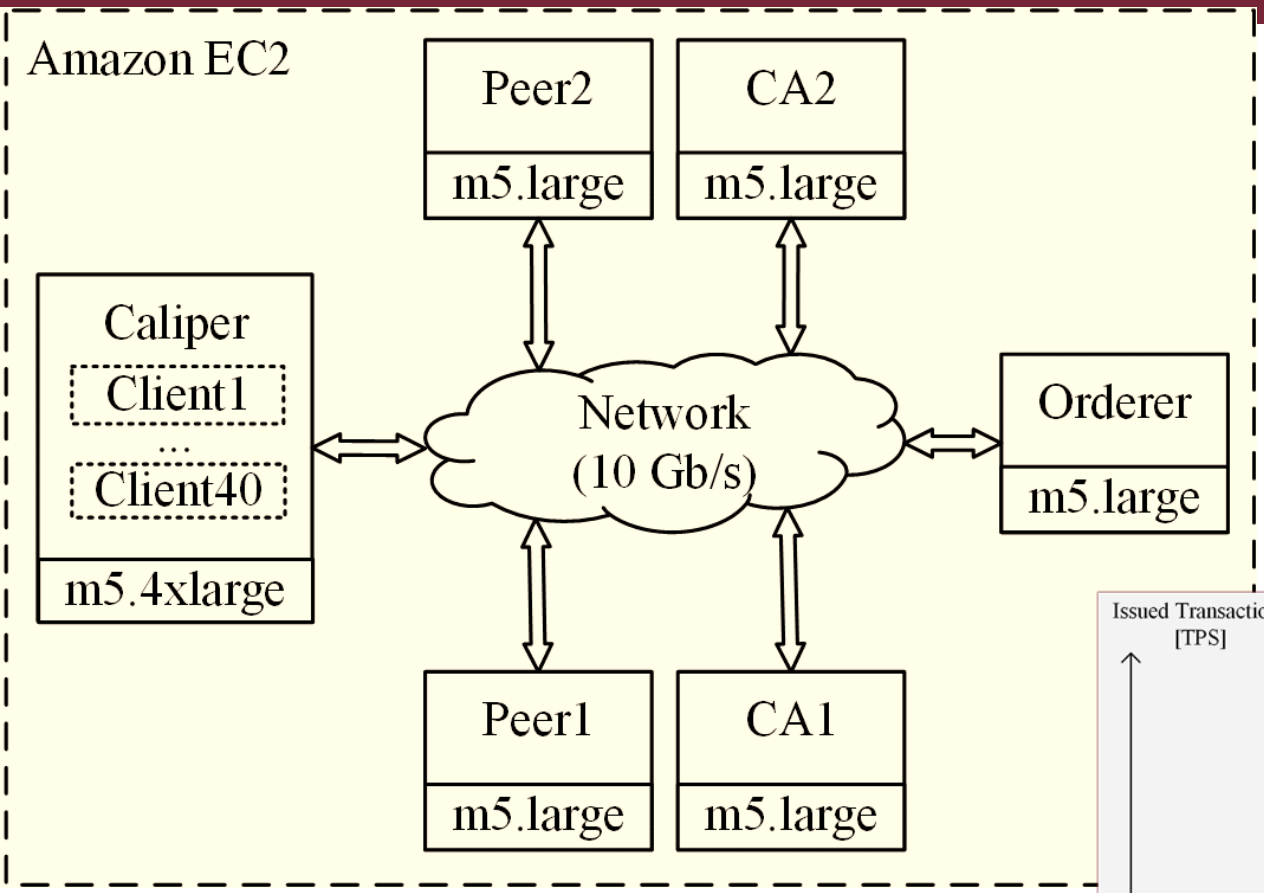
Causal graph

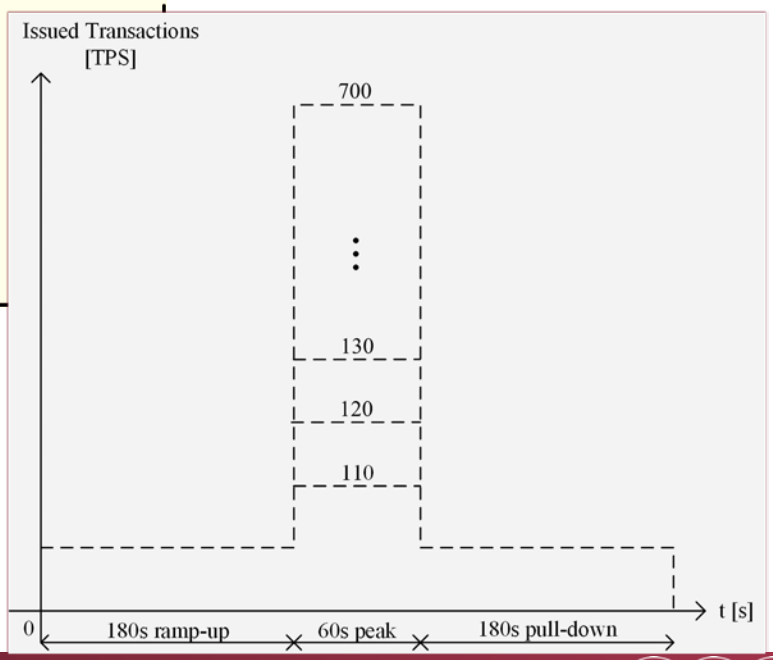
(potential root causes(



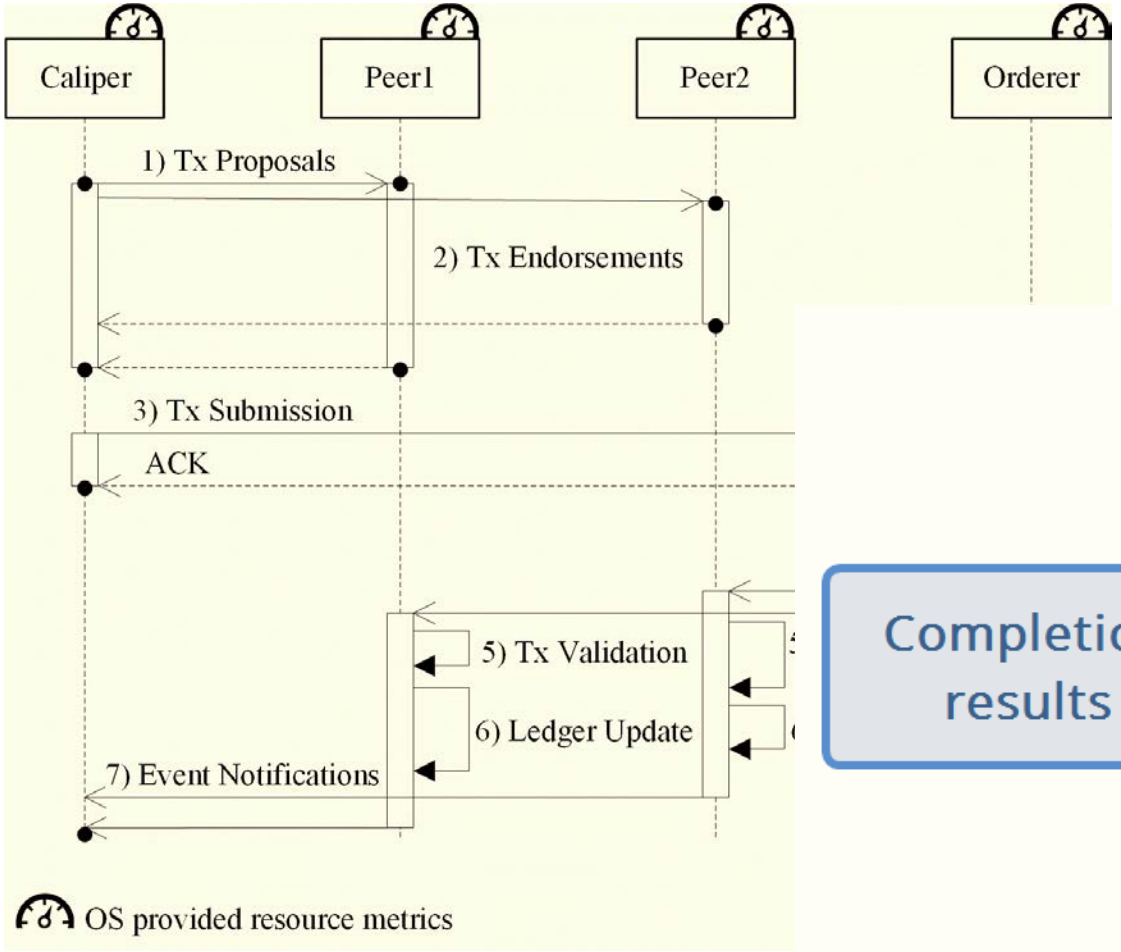
A STRESS-LIKE TEST

Measurement setup

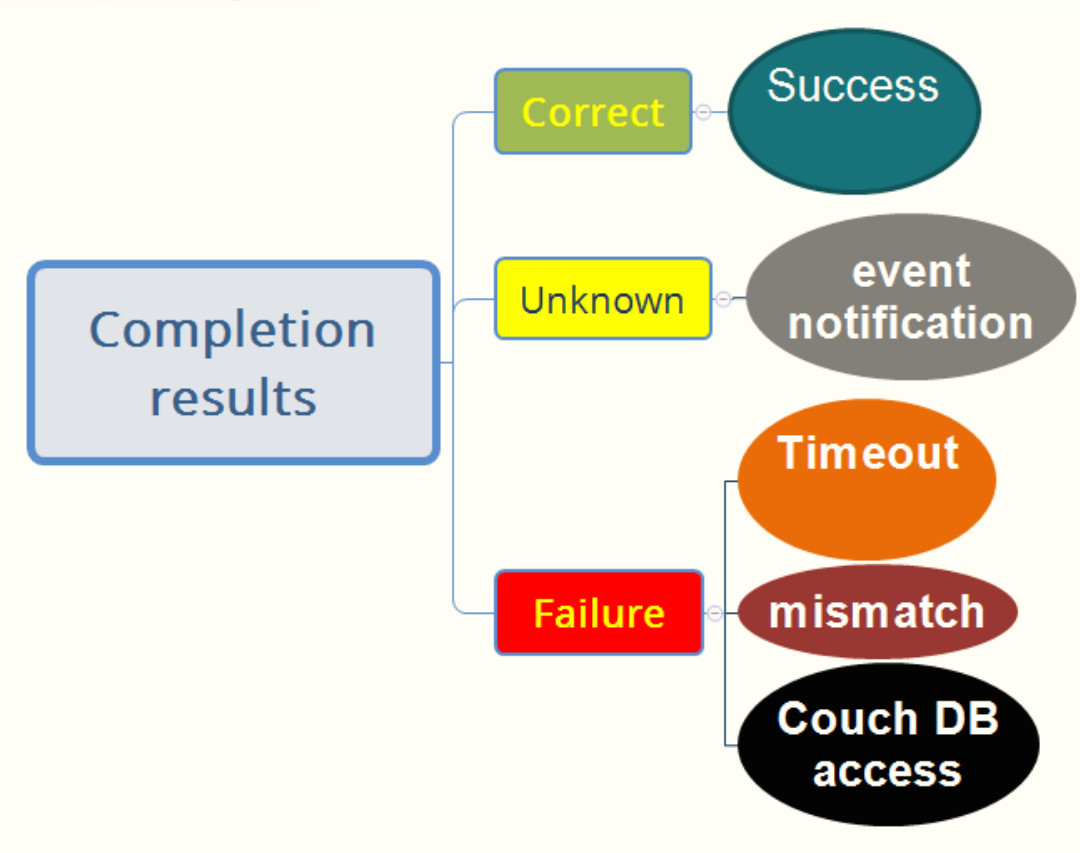




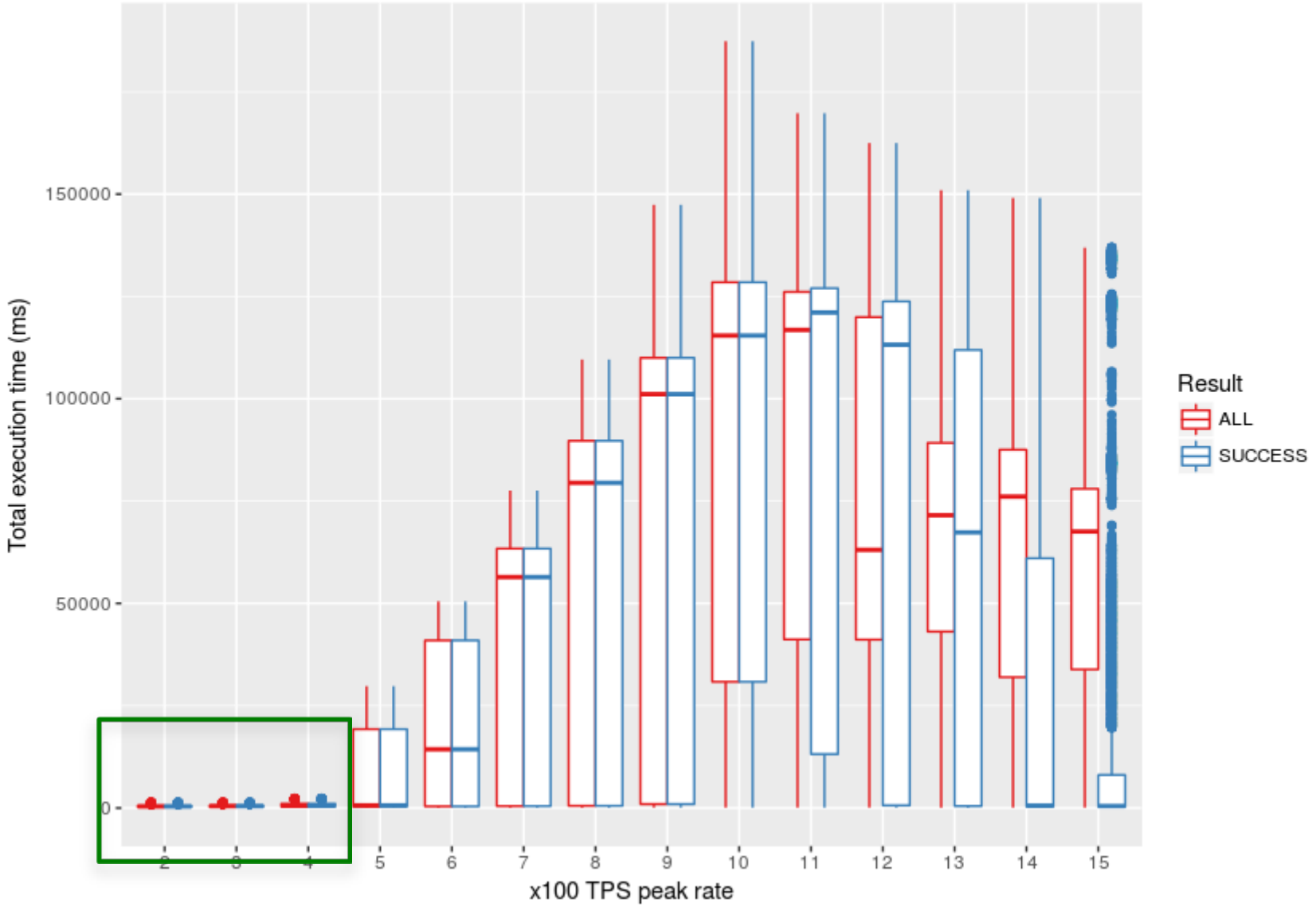
Measurement and outcomes



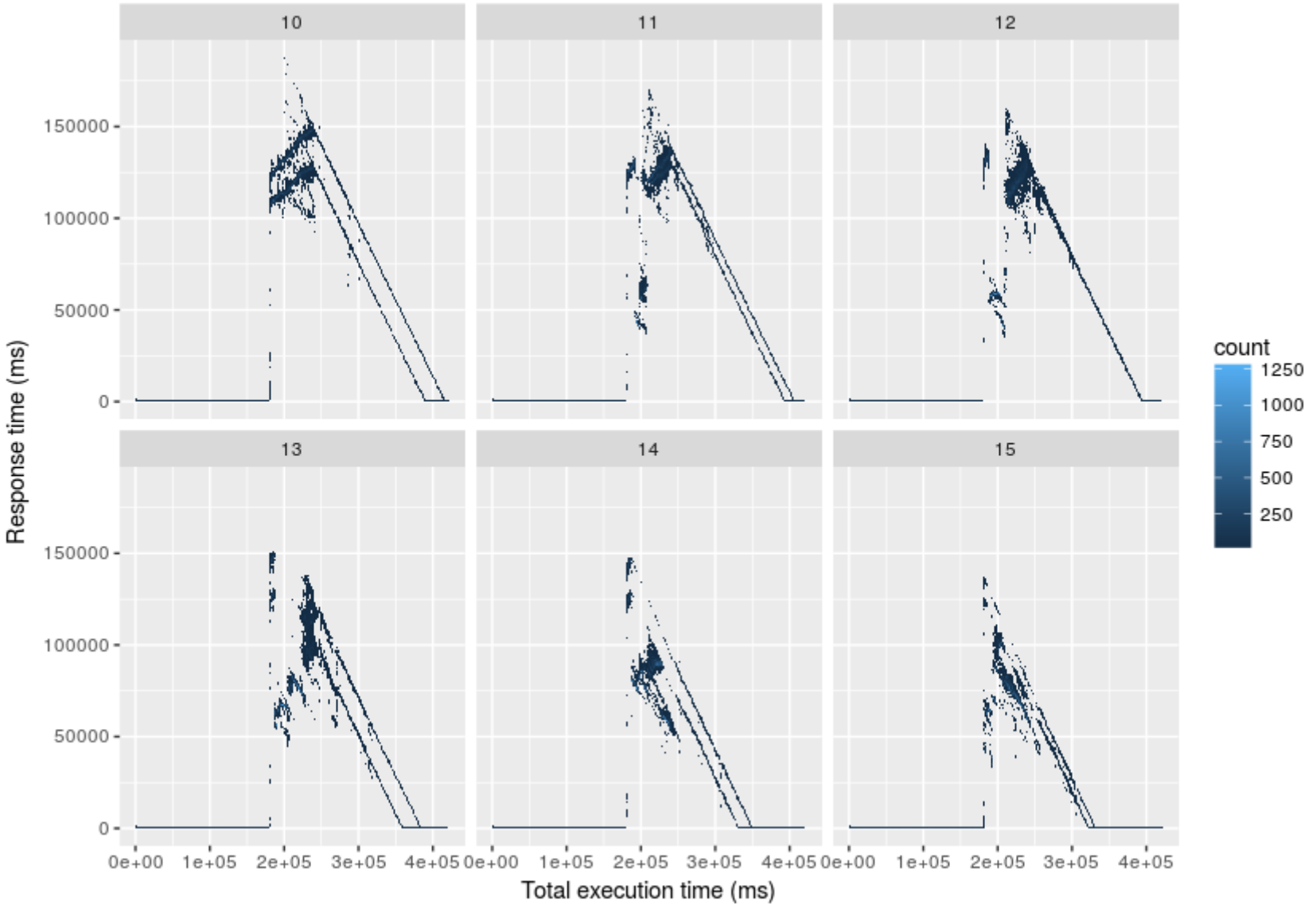
DEMO



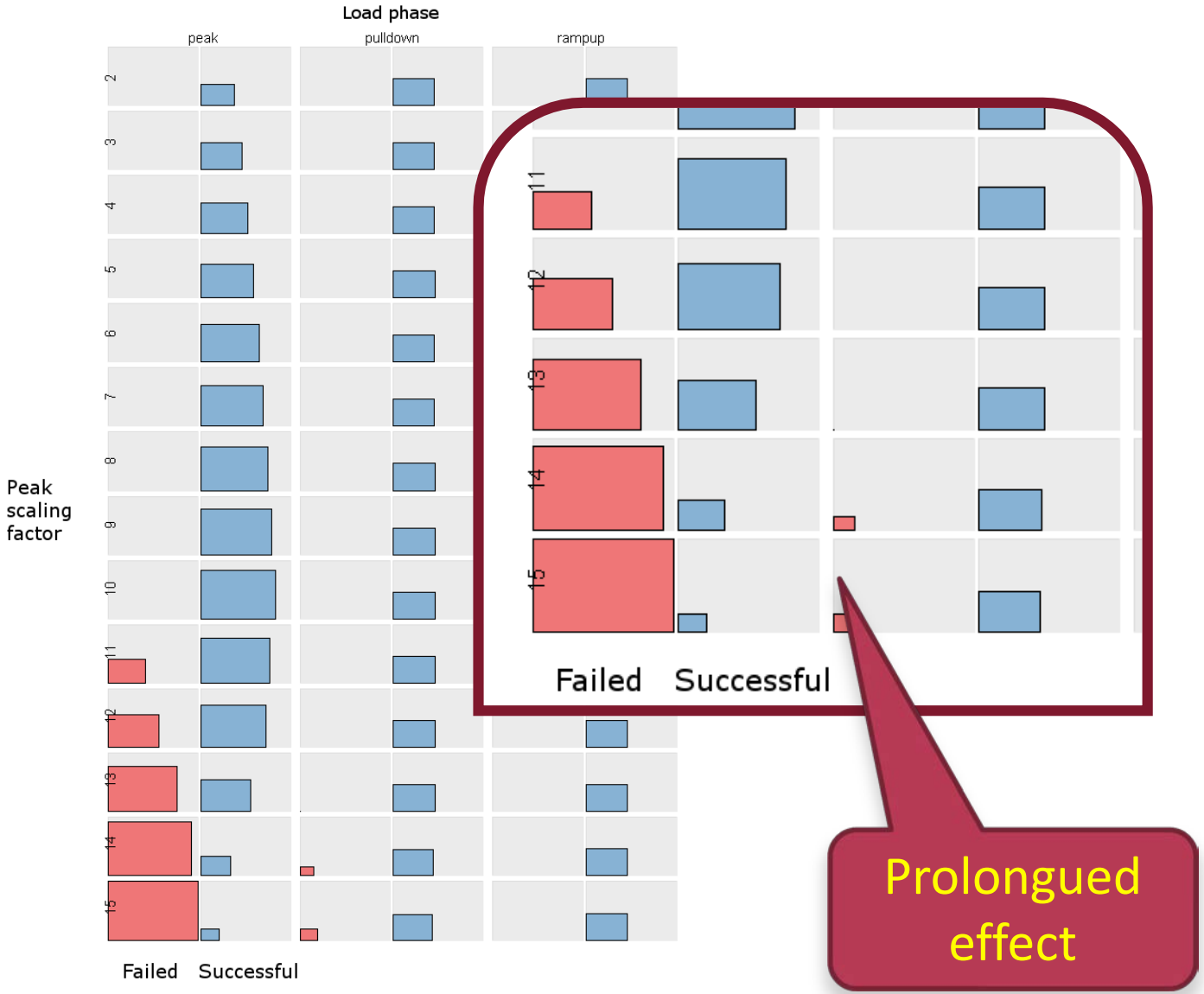
Peak rate to total execution time



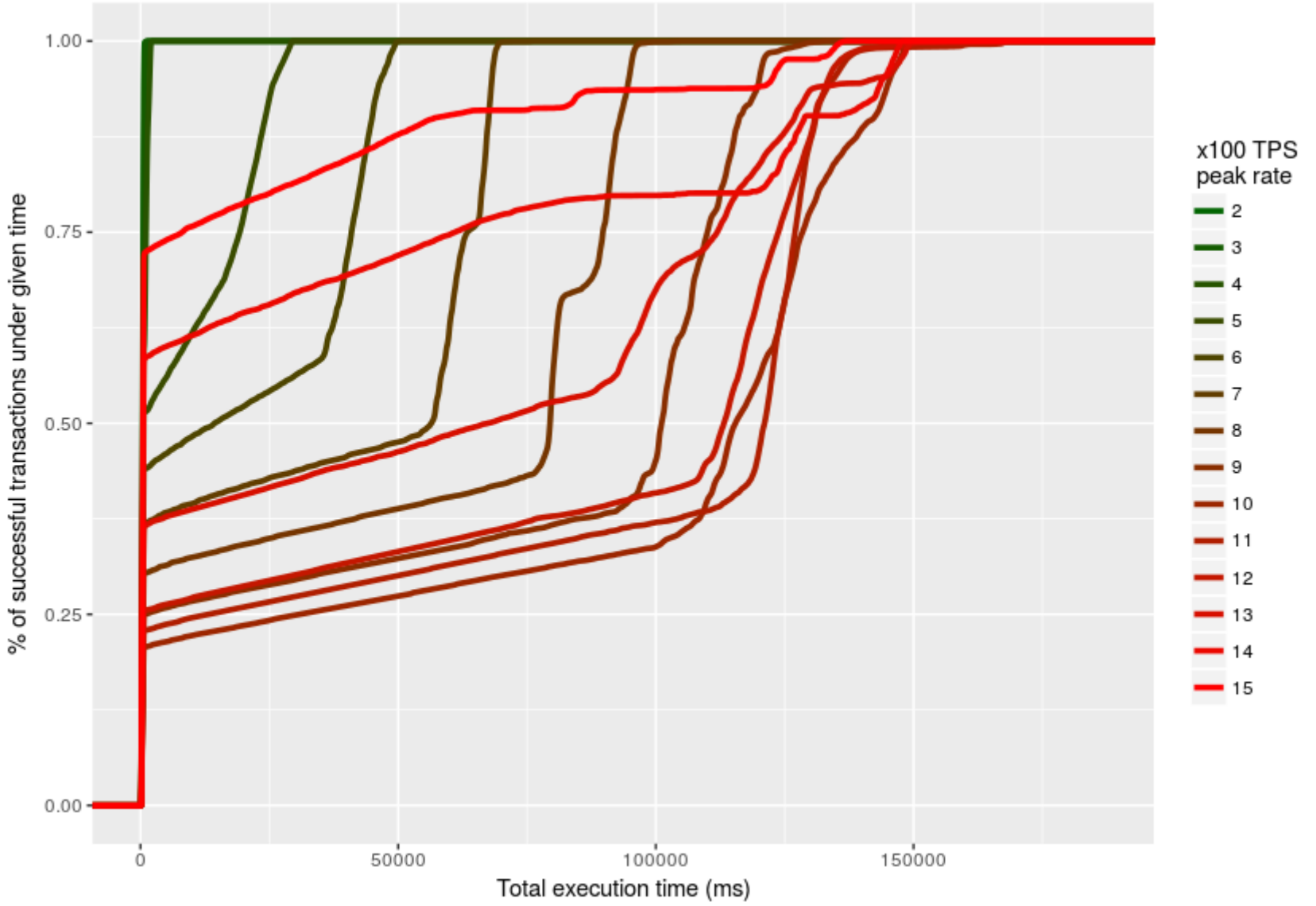
Short (?) transient increase in response time



Failure distribution



Decline of success under high workload

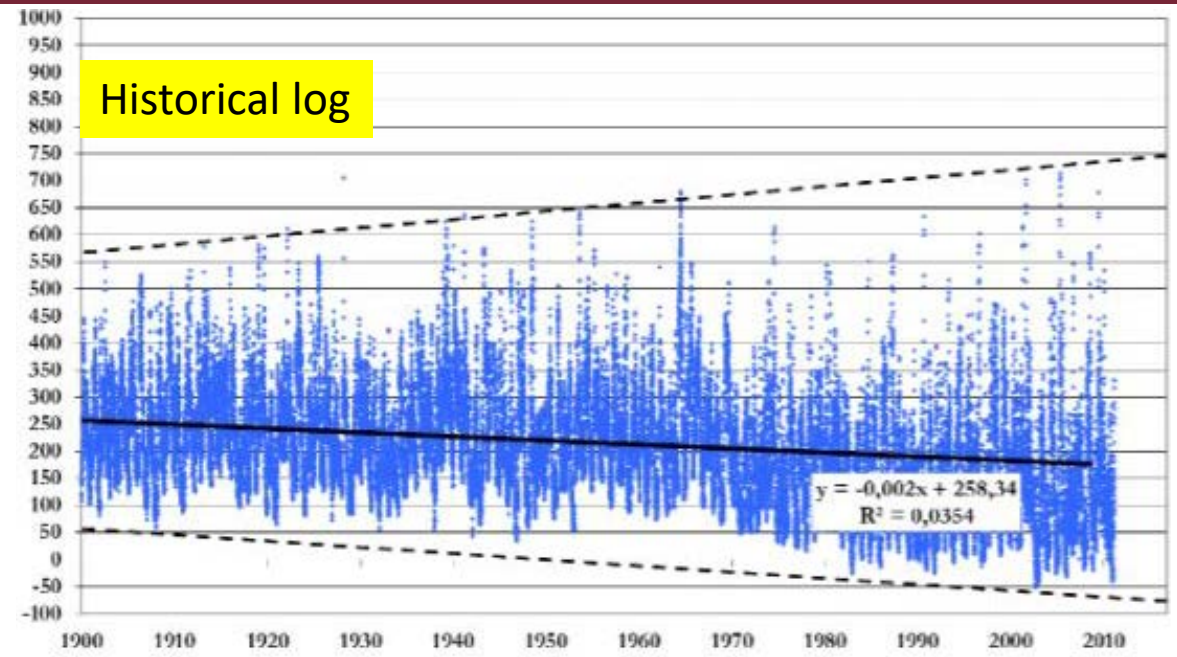


EXTREME VALUE ANALYSIS

Statistical characterization of minima/maxima

...

Level of the Danube by Nagymaros



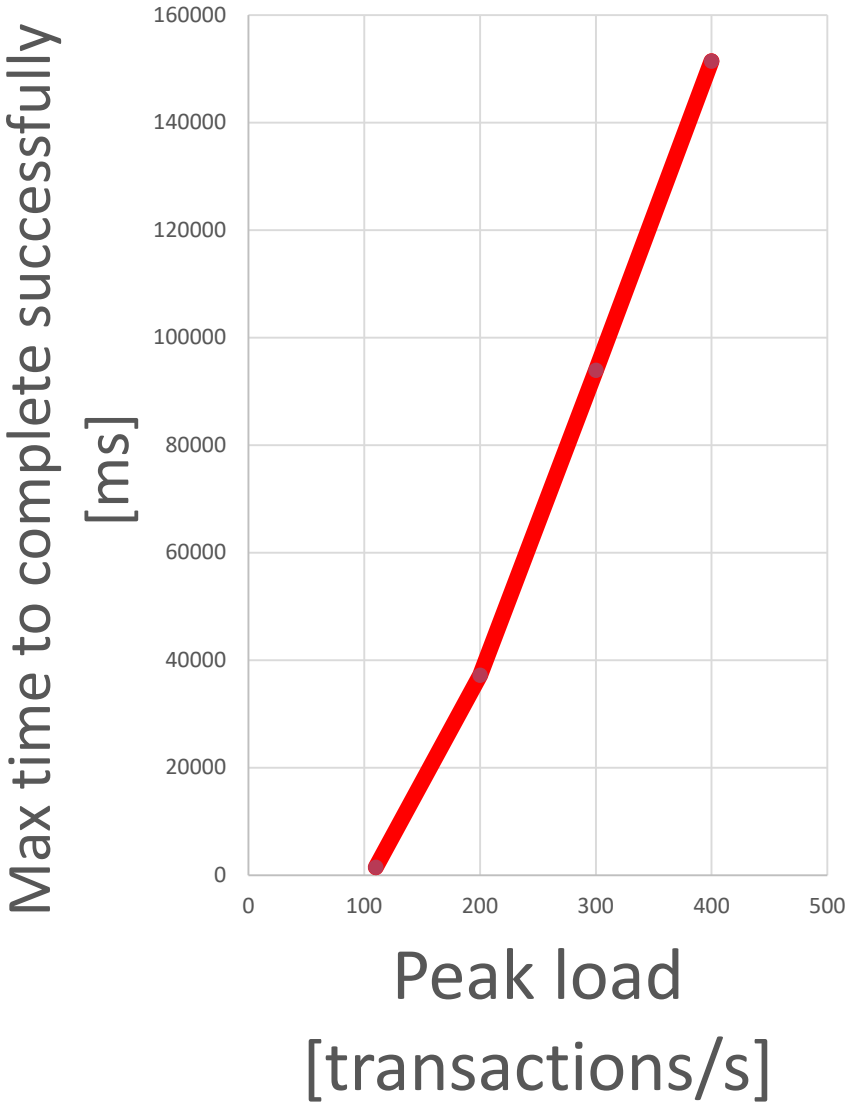
Is the new dam high enough?

Shall I pay extra +15% insurance fee for the high flood risk?

Zs. Nagy-Kovács, K. Dr Takácsné Prof. Dr. György: A Duna Vízszint-Változásainak Vizsgálata Nagymaros és Budapest Vonatkozásában

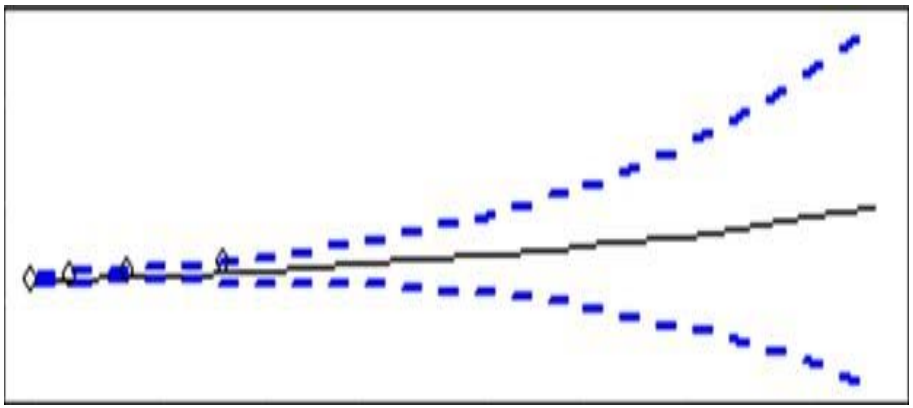
Paper 30 Proceedings of 8th International Engineering Symposium at Bánki [PDF] (ISBN: 978-615-5460-95-1), 2016

Maximum execution times under load

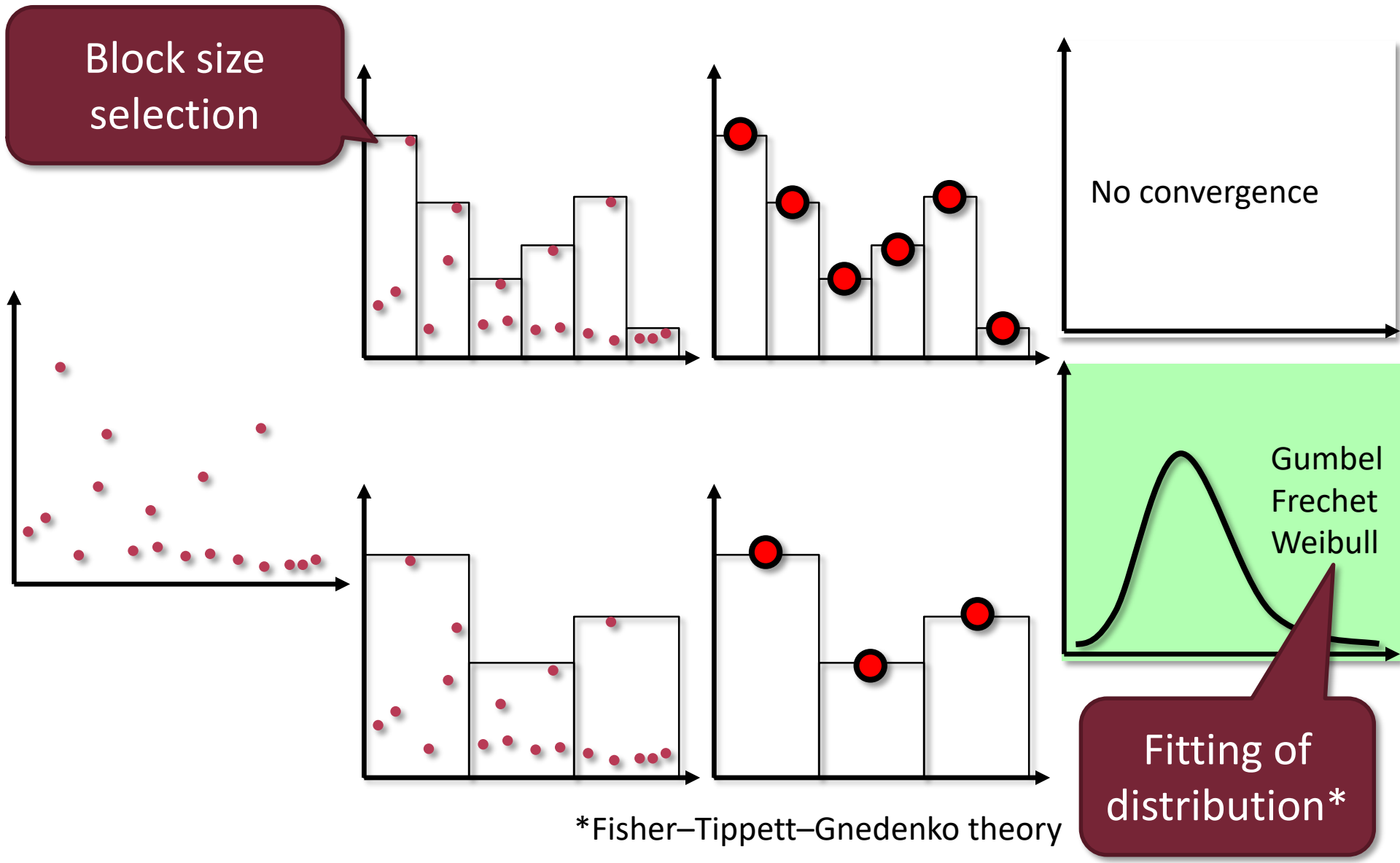


- Workload → maximum execution time
- High probability of

- Expected maximum over time

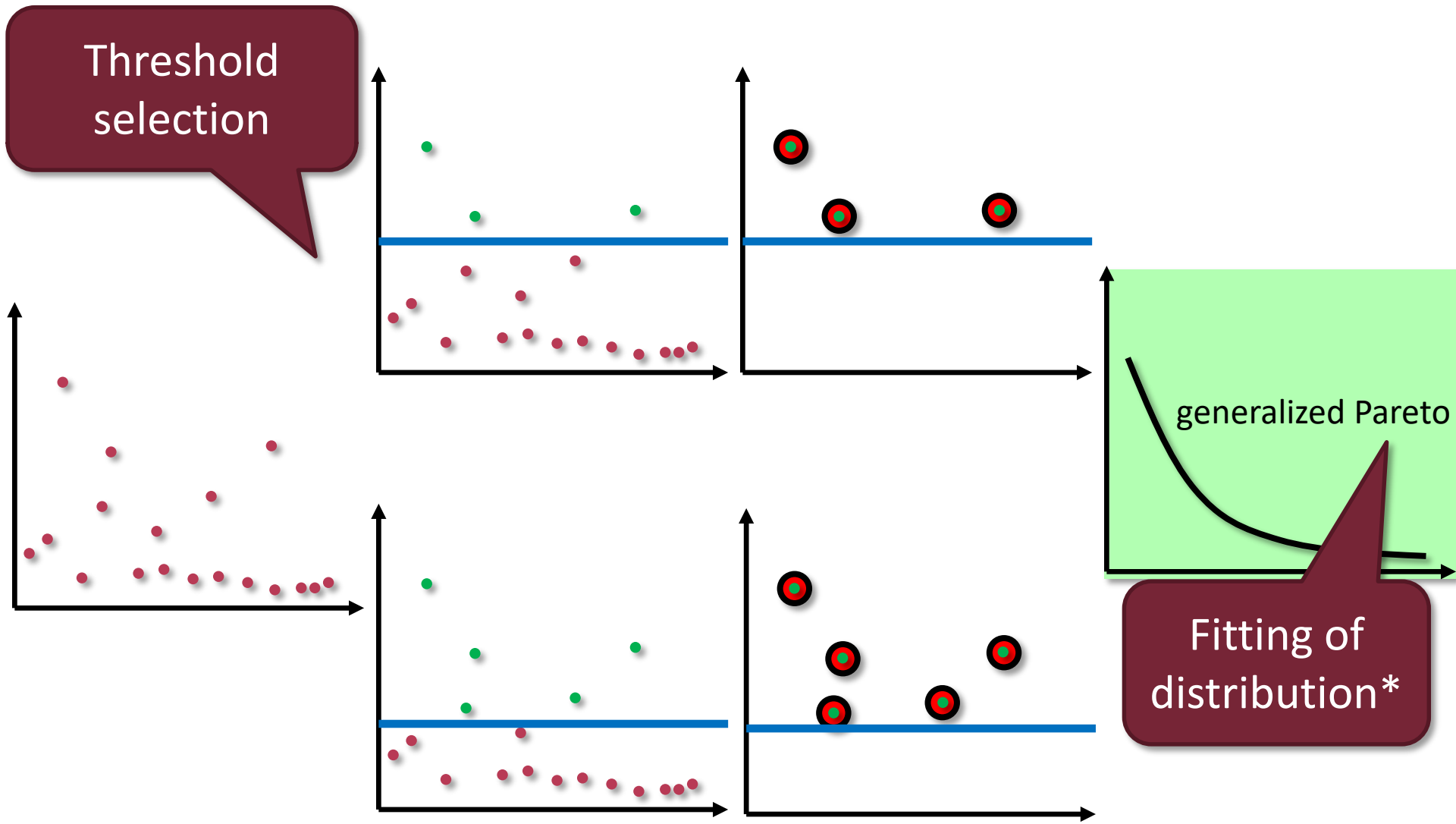


Block Maxima (*Fisher–Tippett–Gnedenko*)



*Fisher–Tippett–Gnedenko theory

Peak over threshold (*Pickands–Balkema–de Haan*)

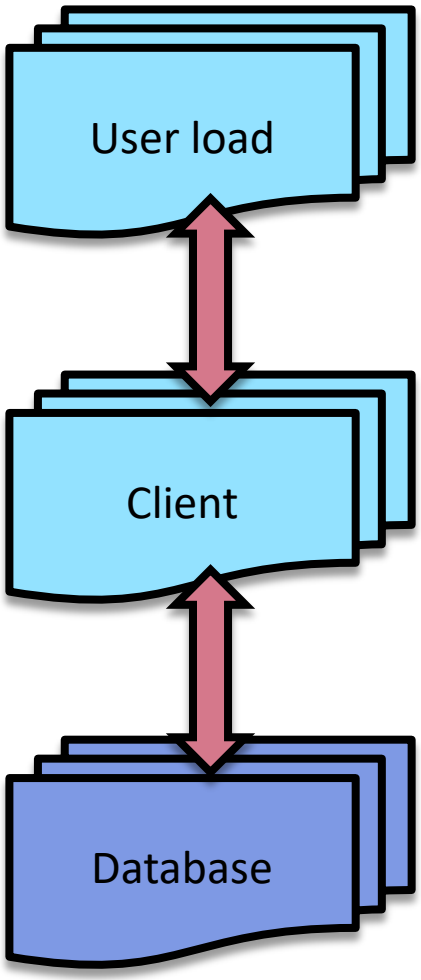


*Pickands–Balkema–de Haan theory

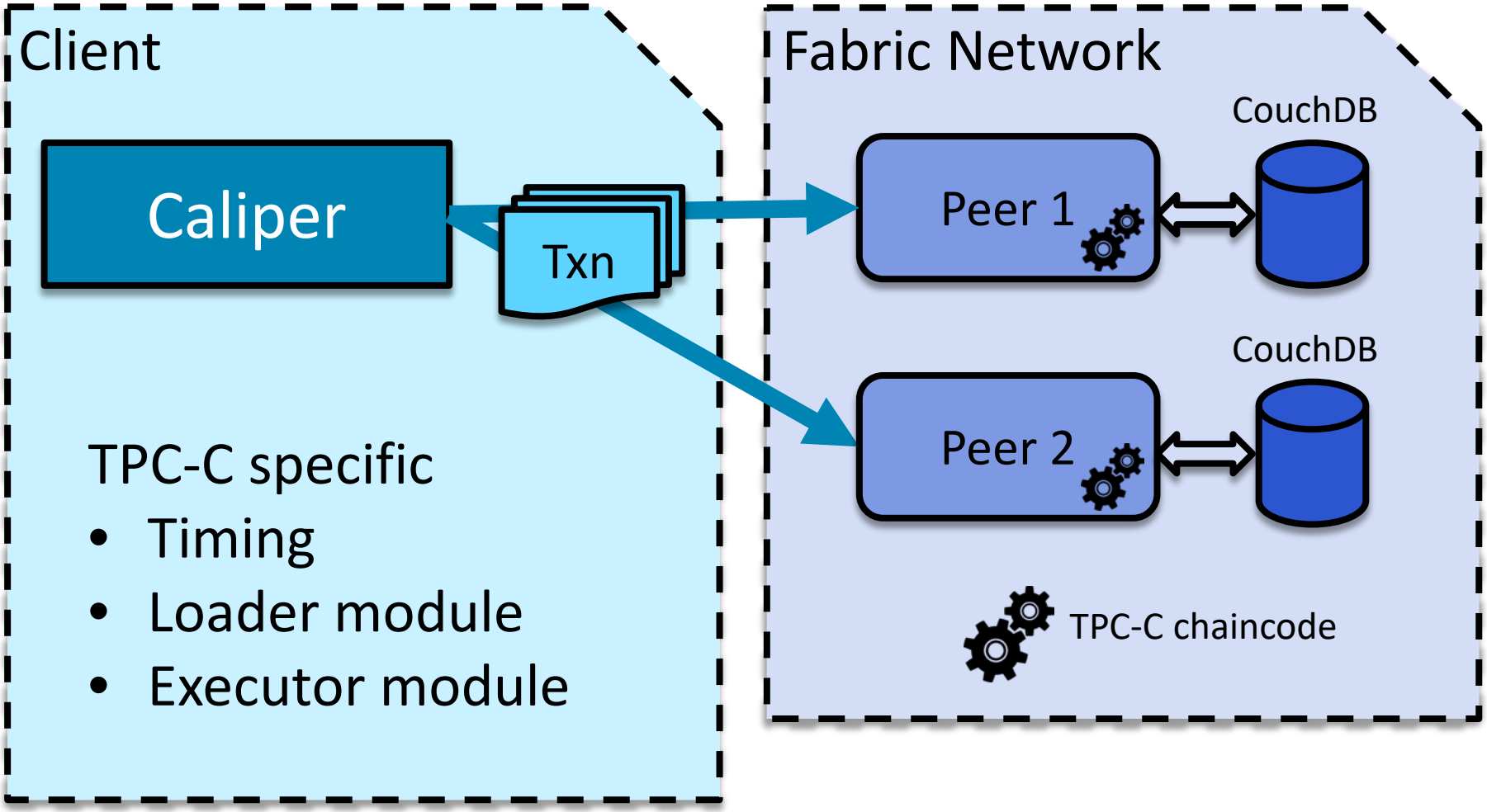
A COMPLEX USE CASE

TPC-C Benchmark

- **On-Line Transaction Processing Benchmark**
- **For traditional databases**
- **Complex database schema**
  - **9 types of tables**
  - **wide range of record and population sizes**
- **5 concurrent transactions**
  - **Different types**
  - **Different complexity**
- **Transactions per minute**



TPC-C Blockchain Benchmark Architecture



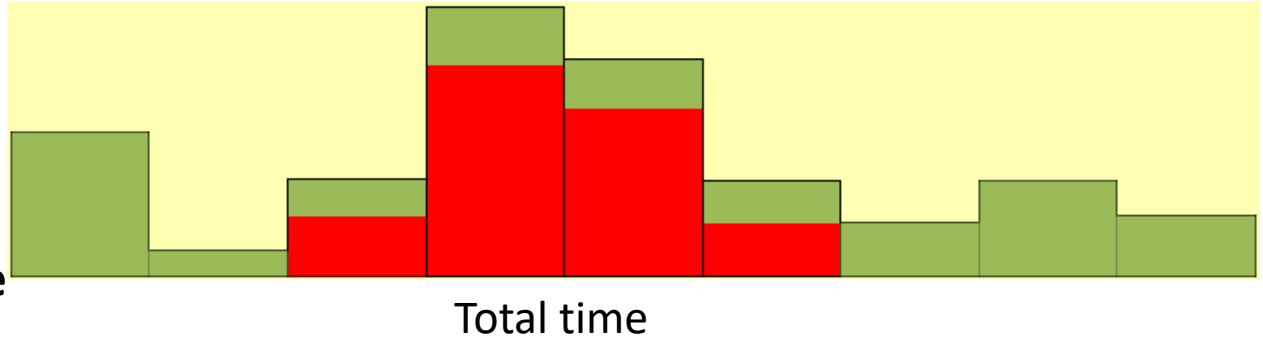
Chaincode error propagation

- **Unhandled error in chaincode**
  - Incorrect query
  - Database timeout

```
[shim] handleQueryStateClose -> ERRO 003
[shim] handleGetQueryResult -> ERRO 004
[shim] handleQueryStateClose -> ERRO 005
```

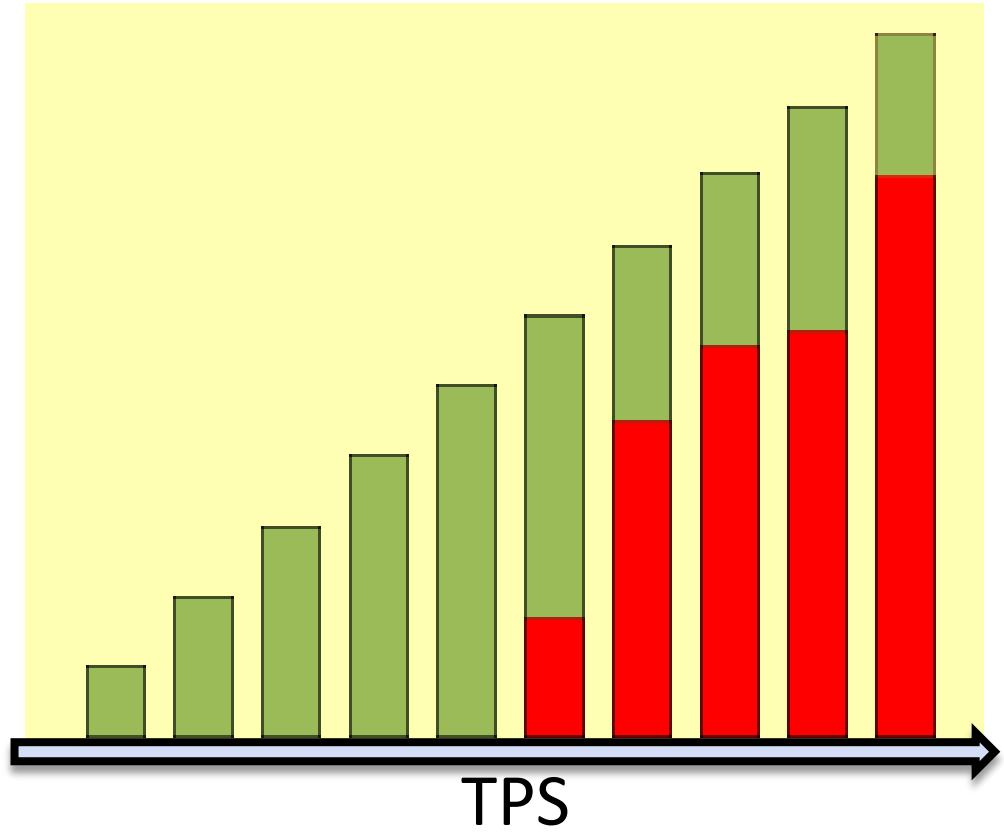
- **Leads to**
  - Propagating faults
  - Unavailable data
  - High verification time

- **Effect**
  - Failed transactions
  - High response time



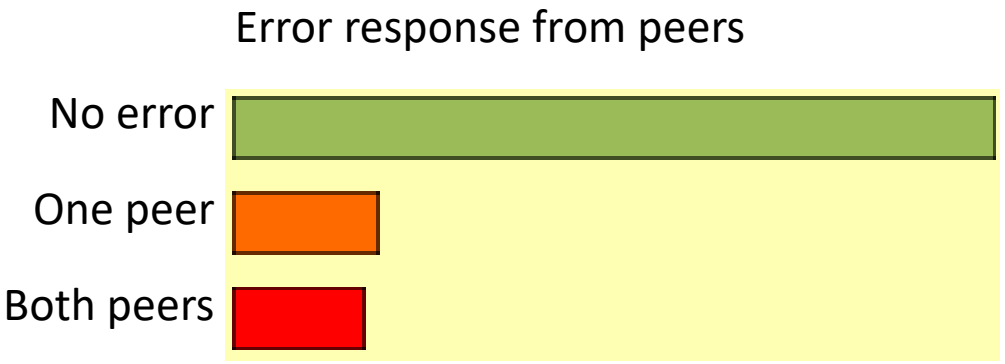
Benchmark

- **Network**
  - **1 Orderer**
  - **2 Organizations**
    - **1-1 Peer + CouchDB**
    - **1-1 Ca**
- **Chaincode**
  - **TPC-C Benchmark (Fabric+Caliper implementation)**
- **Duration: 60s**
  - **1 TPS – 10 TPS**
    - **Growing transaction number**

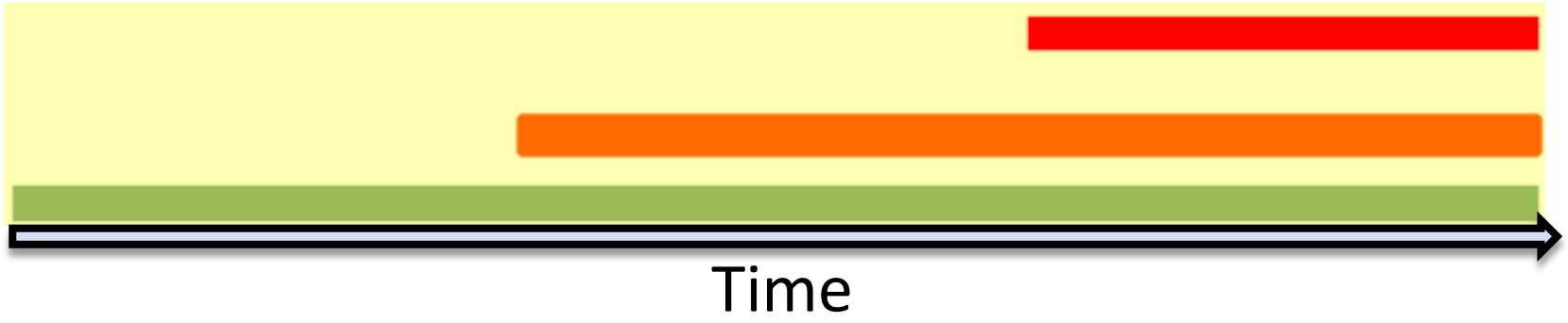


Cause

- **Peers have independent**
  - **transaction execution**
  - **database access**



- **Failure of one peer**
  - **crashes**
    - **the other peer**
    - **whole network**

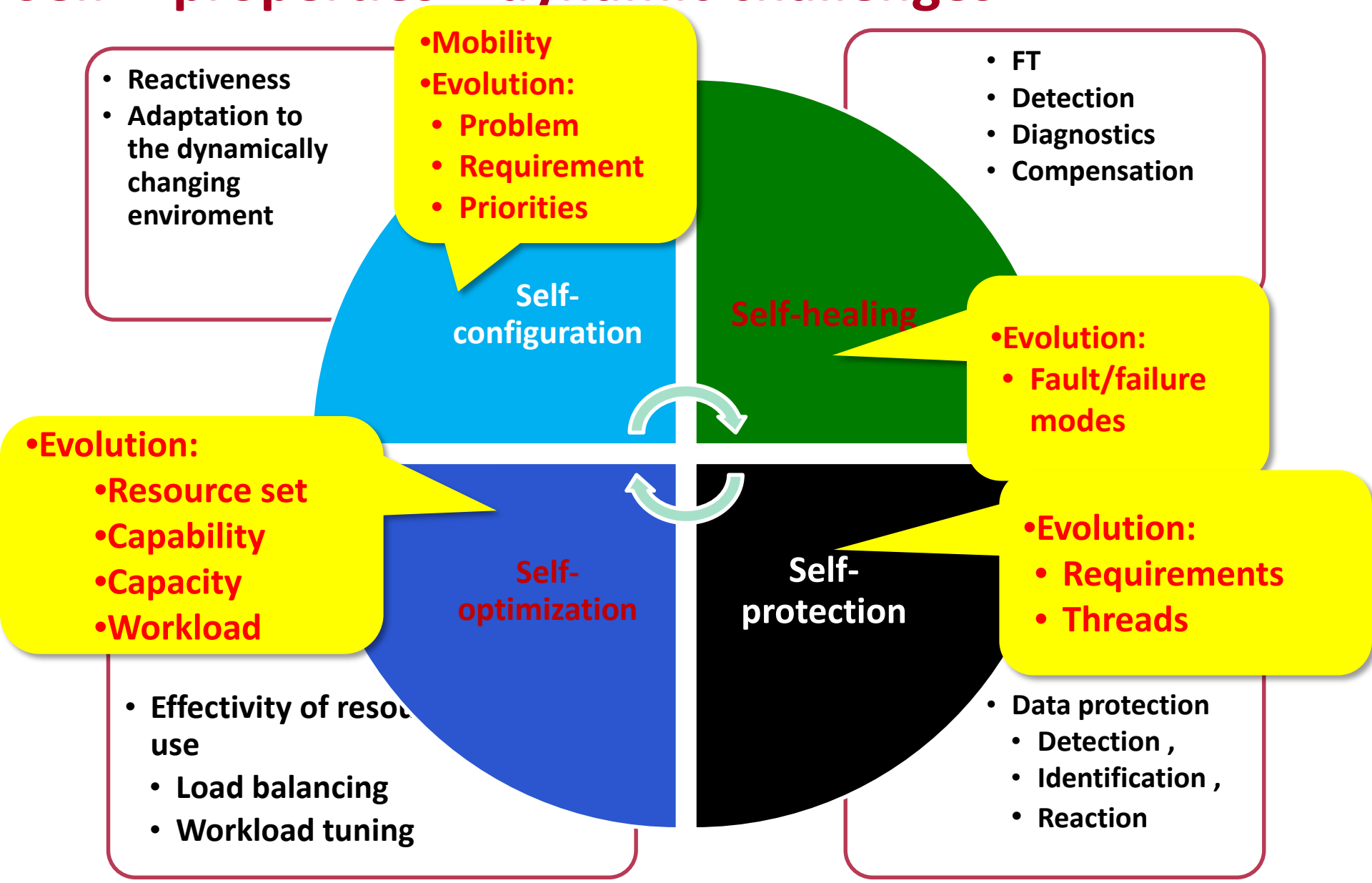


CLOSING REMARKS

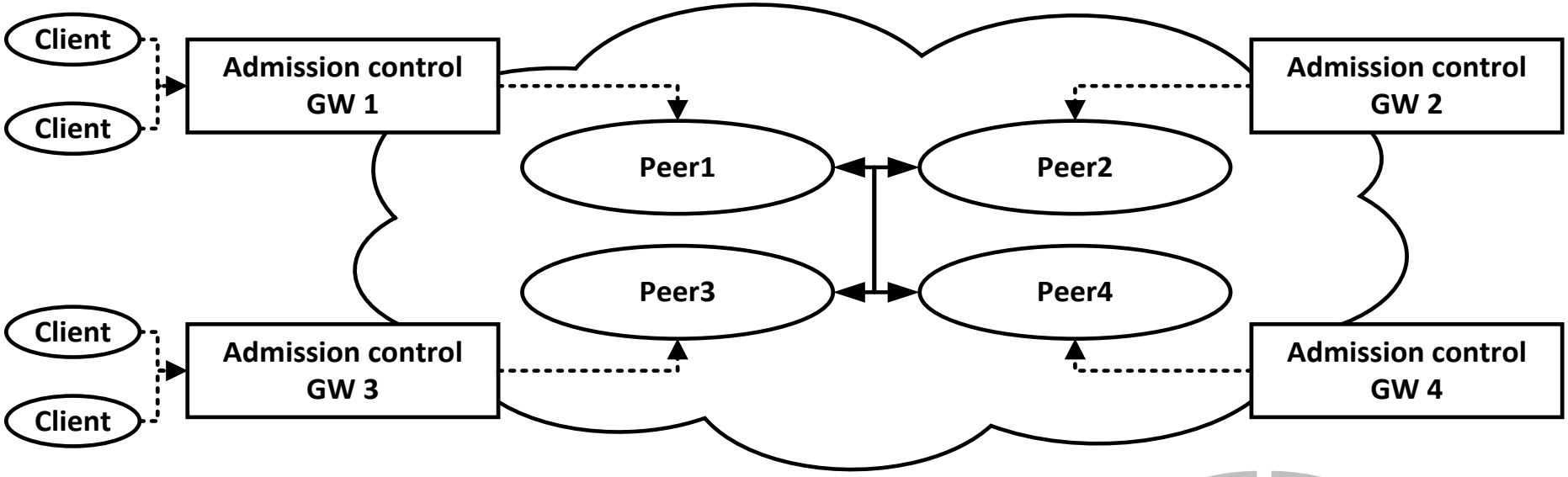
Hyperledger is a promising technology,

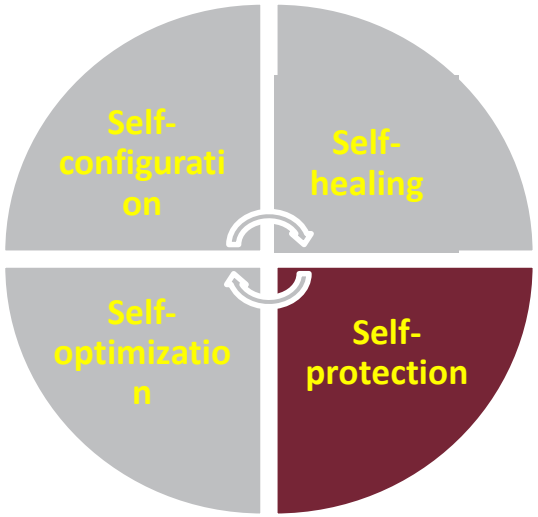
but critical application need a complete ecosystem

Self-* properties – dynamic challenges



PoC: static, cooperative admission control

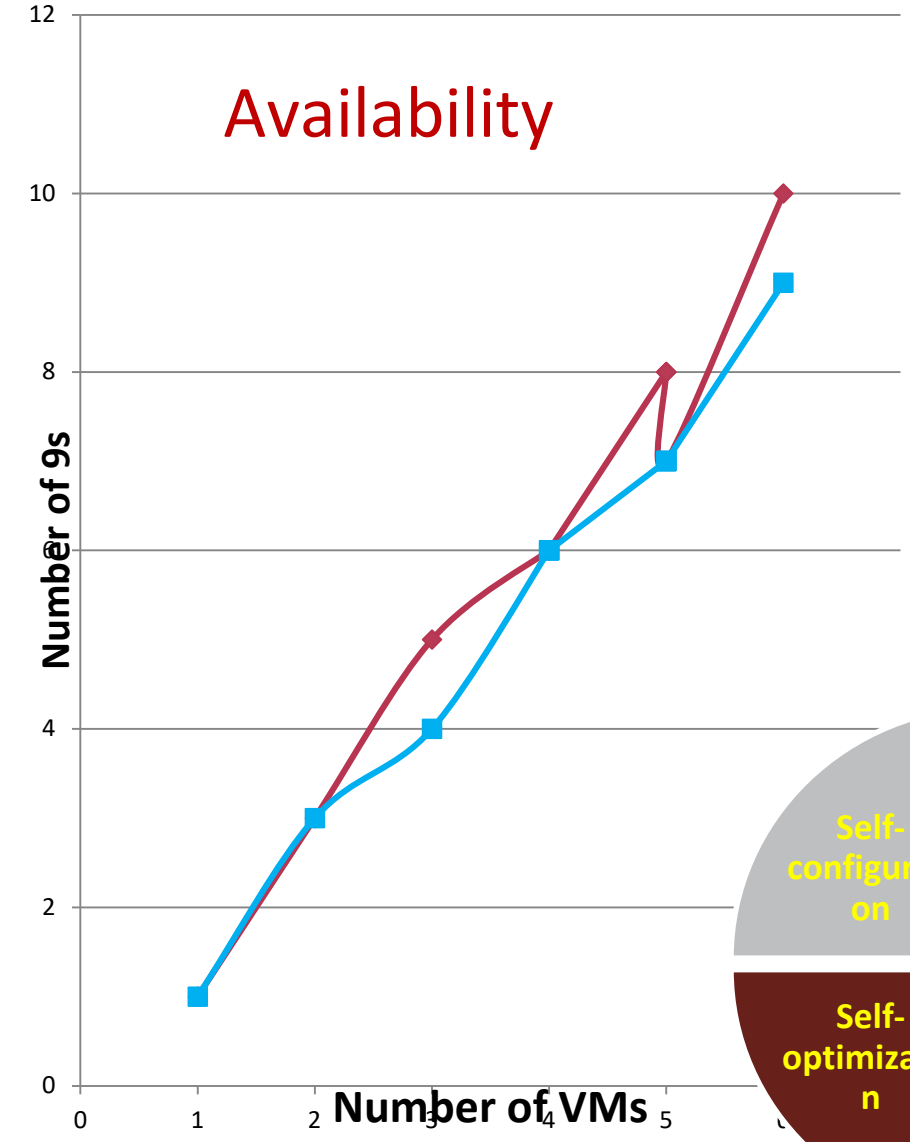


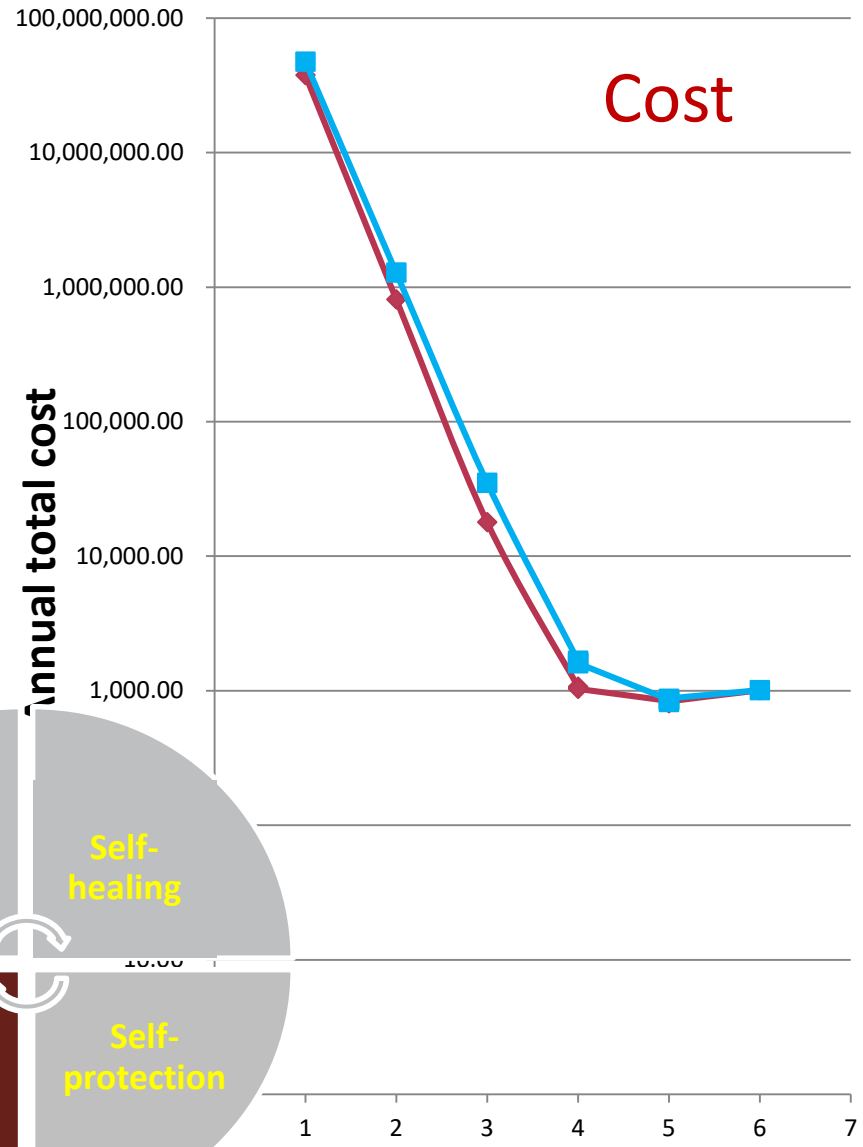


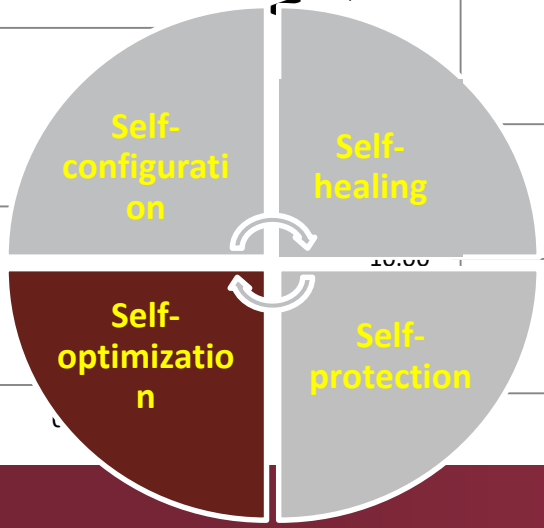
GWs have a capacity "slice" and monitor job completion

First iteration

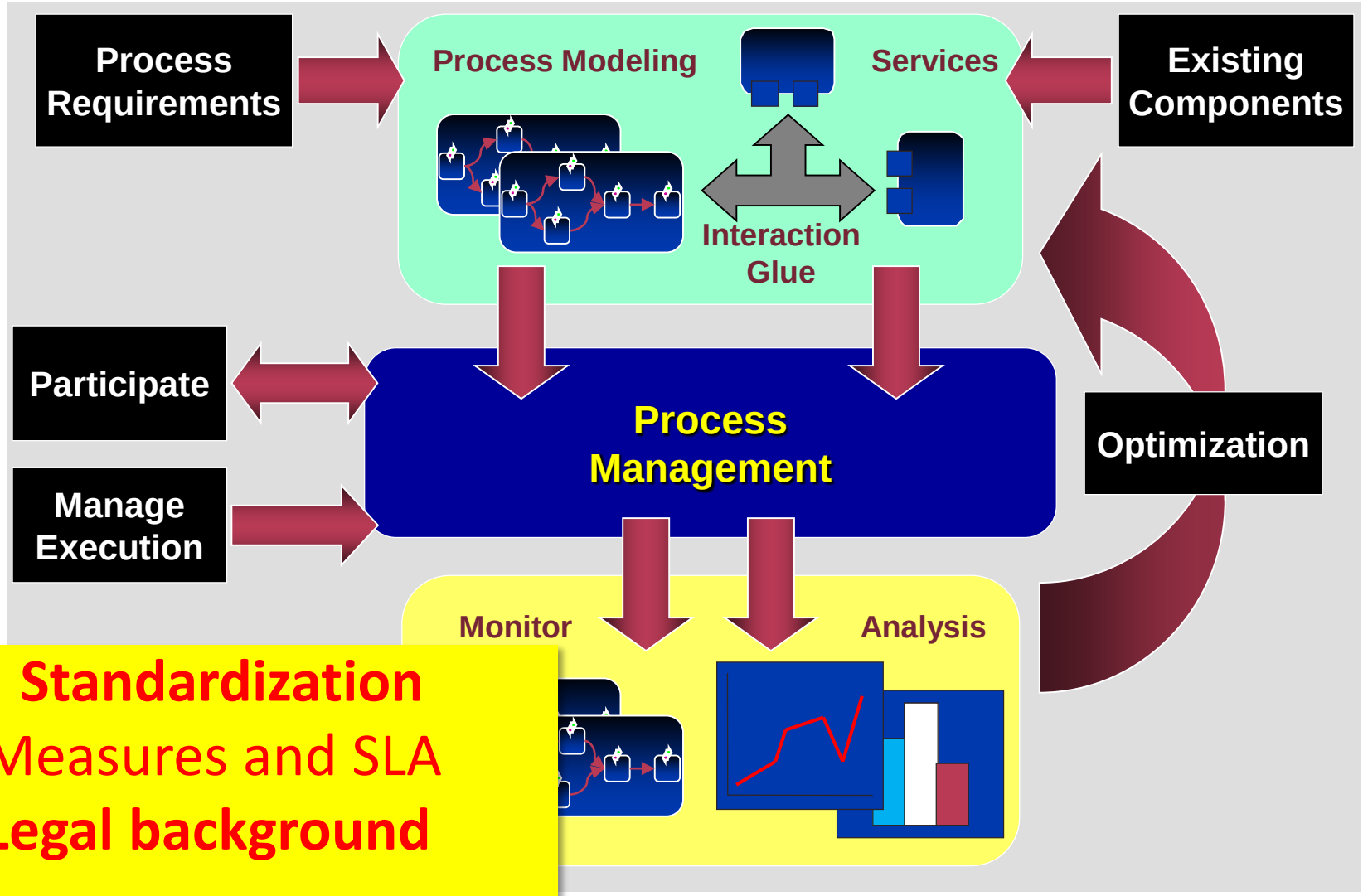
Opportunity: cheap VM based redundancy







Process Development Lyfecyle



Standardization

- Measures and SLA
- **Legal background**

Is it the same old story?

- Distributed vs. time
- Asynchronous operations
  - Delay
  - Timeout
  - Timeliness
- Monitoring and control
- Distributed consensus based admission