Blockchain interoperability

—

Cross-chain oracles with Decentralized Computing

Matthieu BETTINGER

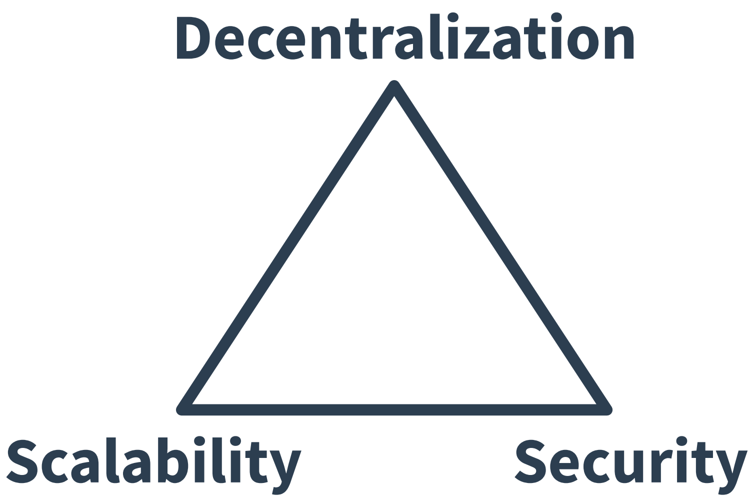
Summary

- Context & Motivation
- State of the Art
- Proposed solutions
- Results & Evaluation
- Conclusion & Future work

Context & Motivation

Blockchain trilemma:

– No "silver bullet"



- **Complex user needs**

  → Integration of distinct blockchain solutions

  – Oracles linking blockchains

- **User needs apply on oracle specs as well**

  → Need flexible oracle architectures

Context & Motivation

- **Generic oracle use-cases**
  - Data queries (request-handling)
  - Triggers (event-handling)

  → Cross-chain oracles
  - Cross-chain calls → Cross-chain smart contracts
- **Non-functional properties**
  - Data integrity & availability
  - On/Off-chain verifiability
  - Scalability (throughput, costs)

Context & Motivation

- **Variety of use-cases and non-functional requirements by end-users**
  - For blockchain systems
  - For oracle systems

  → Use of decentralized cloud computing infrastructures for flexibility

State of the Art

- **Datafeeds:**

  → Feed data to blockchains

+ Support all oracle use cases

– Rigid architectures

State of the Art

- Cross-chain bridges

  → Relay information between chains

+ Scalability, data integrity & availability

– Specialized in requested generic data transfers, no user-defined logic

– (usually) Single data acceptance mechanism

State of the Art

- **Decentralized cloud computing infrastructures:**

  → Execute arbitrary provided logic: a decentralized app

  + Sandboxed architectural freedom

  – Administrative & Genericity-related overheads

- **Objective:**

  → Enable flexible datafeeds & bridges with generic DCCI

Context & Motivation

- **Generic oracle use-cases**
  - Data queries (request-handling)
  - Triggers (event-handling)

  → Cross-chain oracles
  - Cross-chain calls → Cross-chain smart contracts
- **Non-functional properties**
  - Data integrity & availability
  - On/Off-chain verifiability
  - Scalability (throughput, costs)

Research questions

- **Secure data transfers using DCCIs**
  - data integrity, on/off-chain verifiability
- **Flexibility of oracle system properties**
  - data acceptance, scalability, decentralization
- **Cost reduction strategies**

Proposal 1: Proxy tasks

+ ***Cost reduction:***

  → *Blockchain trust algorithm (PoCo) delegated on a fast and cheap chain*

  → *Storage reduction: Only results and traceability metadata stored on the costly chain*

+ ***Code & Data integrity*: TEE to securely generate & transfer results**

Inherited from iExec infrastructure:

– *Usability; Storage costs*: Inefficient untyped blob storage for results

– *Reactivity; Throughput*: Single result per task

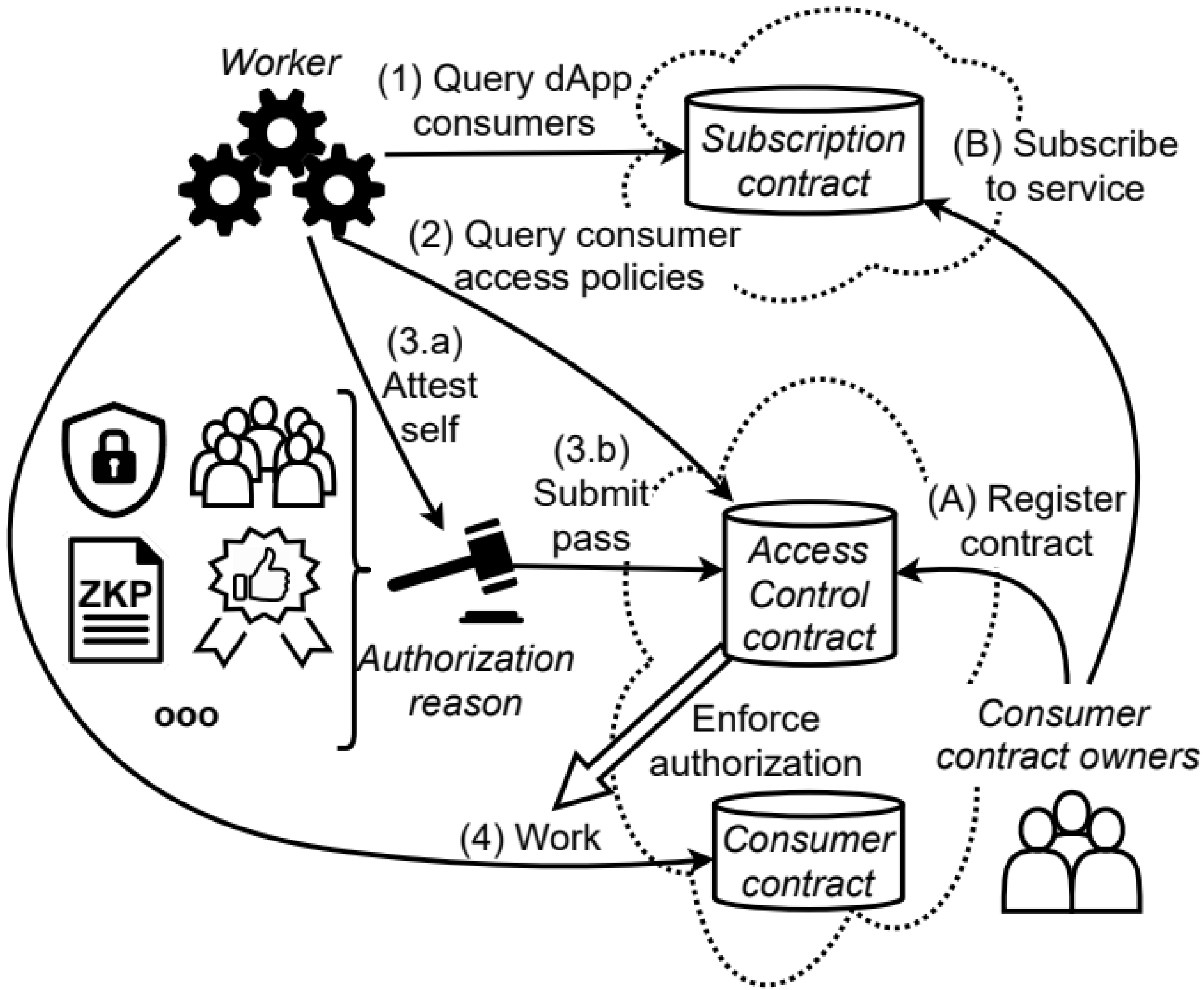
Proposal 2: Work passes
Basics

Access control lists (RBAC):

- A pass links:
  - 1 off-chain entity (e.g. an iExec worker)
  - 1 contract (or a subset of methods)
  - 1 scope of authorization
    - OWN > DELEGATE > WORK > NONE
    - For a limited time

Proposal 2: Work passes Workflow

- **Authorization reason**
  - Attested TEE
  - Committee consensus
  - (ZK) Proof
  - Trusted identity
  - …



Proposal 2: Work passes
Protocol properties

\+ *Reactivity; Throughput*: Multiple results per dApp execution

\+ *Flexible security*: Consumer-defined data acceptance logic

\+ *Usability; Storage costs* : Application-specific typed storage

– *Security risk*: Direct access to consumer methods

→ Intermediate dApp-specific storage as buffer

– *Storage costs*: A pass per consumer contract

Proposal 2: Work passes
Protocol optimizations

- **One-time passes**

    + No persistent storage costs

    – Repeated authorization verification

- **Per-dApp passes**

  → 2 passes on-chain : Contract-Service & Service-Worker

    + Only one pass per worker per dApp per chain

    – Additional storage costs + pass linking costs

Results:
General Metrics

- **Full PoCo task execution with TEE worker :**

  → ~5 minutes of non-productive processing

  Proxy tasks VS current iExec tasks :

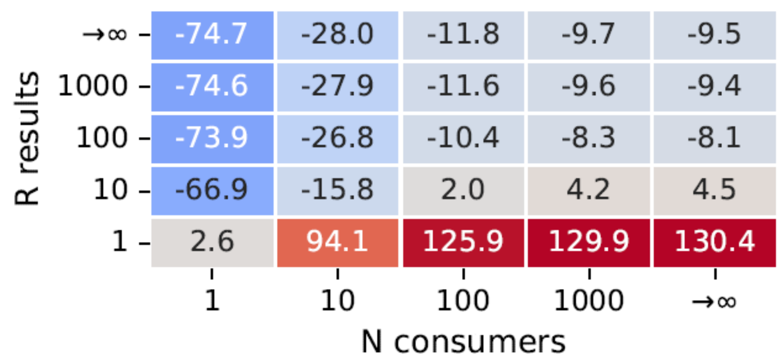
  → –64 % gas costs

Results : Algorithmic complexities

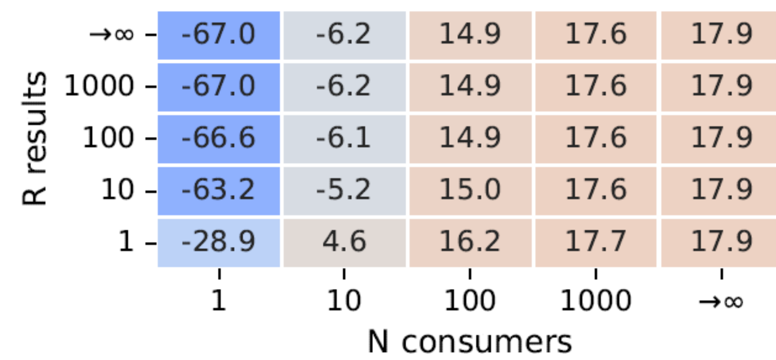
Complexity \ Method	Proxy tasks	Time-bounded work passes	One-time work passes	Per-service work passes
On-chain	$O(R * C)$	$O(C * N)$	$O(C * N)$	$O(C)$
Off-chain	$O(R)$	$O(C * N)$	$O(R * C * N)$	$O(1)$

Table 6.2: Administrative cost complexities
Complexities are relative to the number C of target chains, N consumer contracts per chain (on average), and R results to transfer.

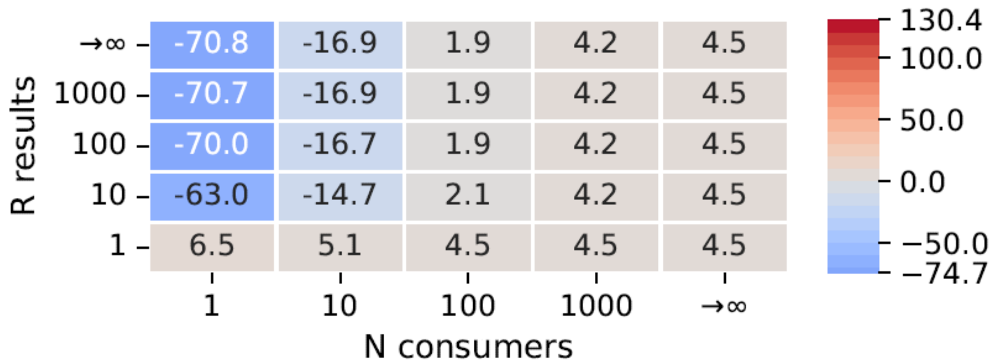
Results: Scaling costs



(a) *Time-bounded work passes*



(b) *One-time work passes*



(c) *Per-service work passes*

$$\lim_{N\to\infty, R\to\infty}\left(\frac{G(N,R)_X - G(N,R)_{PT}}{G(N,R)_{PT}}\right) = \frac{G_{receiveResultX} - G_{receiveResultPT}}{G_{receiveResultPT}}$$

Table 6.3: Relative costs (%) of *work pass* methods compared to *proxy tasks*, depending on the number N of consumers and R results to transfer

Results:
Impact of N-Batching

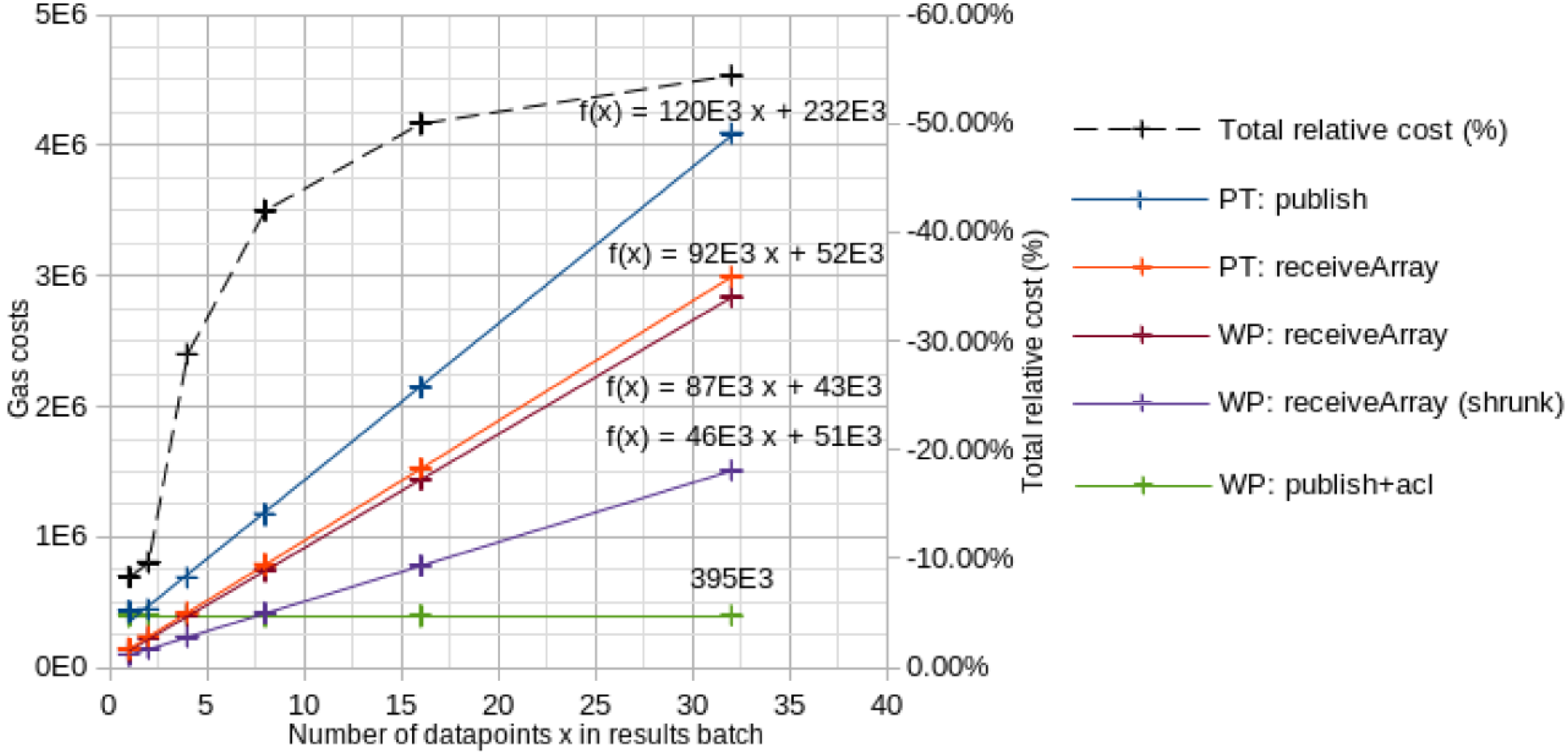


Figure 6.4: Gas costs to transfer n data points at once, using *Proxy tasks* (Proxy) or *Work Passes* (WP)

Conclusion

- **Reuse of trusted security mechanisms**
- **Less storage and processing on costly blockchains**
- **More genericity:**
  - Data acceptance mechanisms
  - Result size and number

Future work

- **Scalability stress tests**
- **dApp modularity/composability**
- **New data acceptance mechanisms**
  - Consensus for streamed results