Duke UNIVERSITY Yale

Perils of Parallelism

Transaction Fee Mechanism under execution uncertainty

Sarisht Wadhwa, Aviv Yaish, Fan Zhang, Kartik Nayak

USENIX Security, 2026

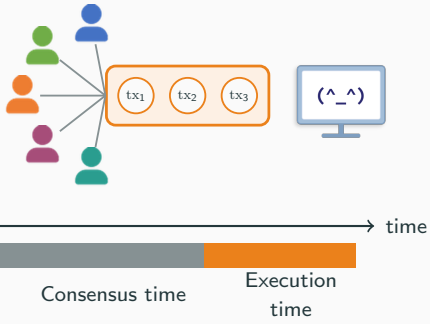
Today's Talk

1. Motivation
2. The Model
3. First Peril: Shill Transactions
4. Second Peril: Uncertain Execution
5. Results and Takeaways

Motivation

The throughput bottleneck has shifted!

Before: Consensus Bottleneck

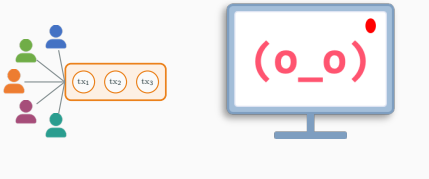


The throughput bottleneck has shifted!

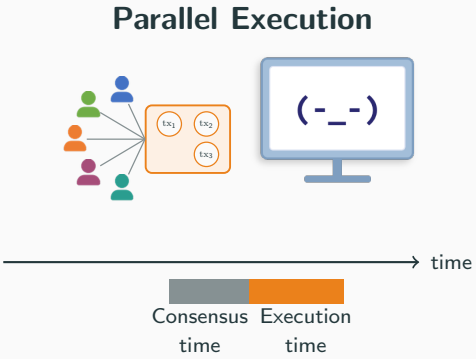
Before: Consensus Bottleneck



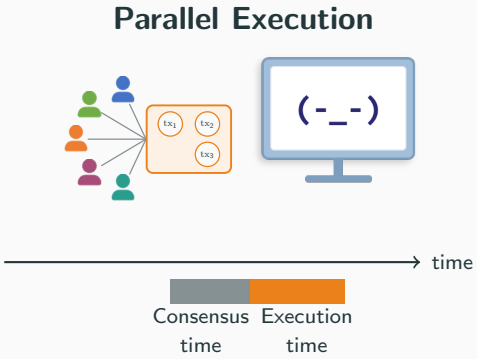
Now: Execution Bottleneck



Solution: Execute multiple transactions in parallel!



Solution: Execute multiple transactions in parallel!



sei

Sui

MONAD

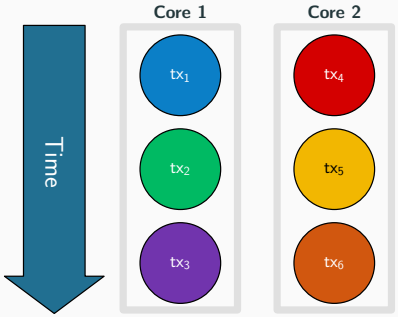
APTOS

ethereum

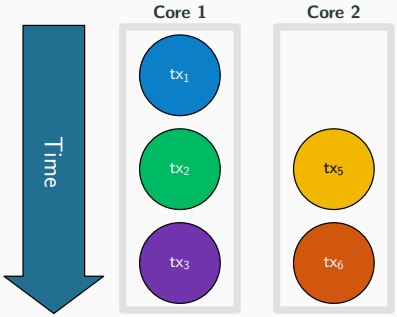
SOLANA

Pricing Parallelism

Case: tx_1 parallelizable

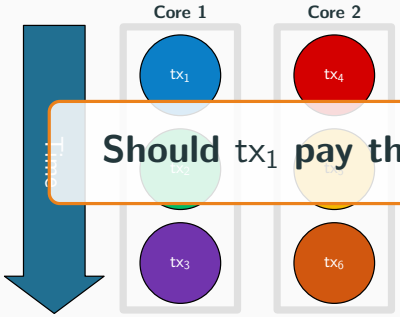


Case: tx_1 not parallelizable

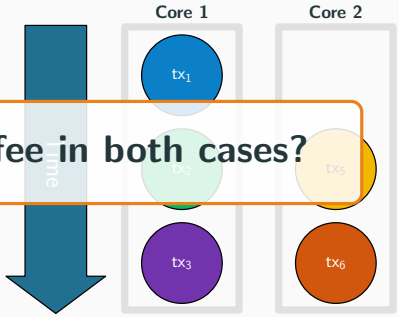


Pricing Parallelism

Case: tx_1 parallelizable



Case: tx_1 not parallelizable



Should tx_1 pay the same fee in both cases?

Pricing Parallelism

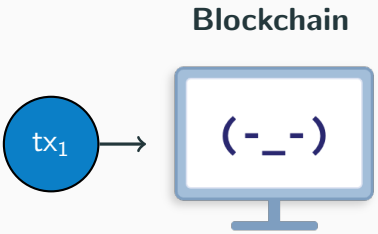


Transaction Fee Market Design for Parallel Execution

Bahar Acilan[1]
ETH Zurich
Andrei Constantinescu
ETH Zurich
Lioba Heimbach
ETH Zurich
Roger Wattenhofer
ETH Zurich

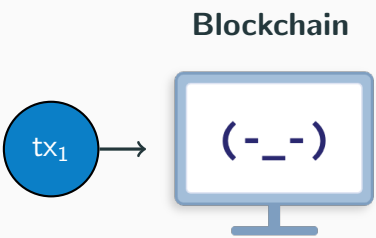
The Model

Objects and Transactions



Object: Data that any transaction reads, modifies, or creates.

Objects and Transactions

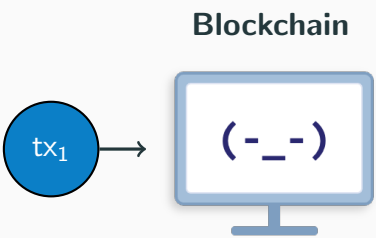


Object: Data that any transaction reads, modifies, or creates.

$$\mathsf{tx} := (t, g, \pi, \{R, W\})$$

- t: Compute time (upper bound)
- g: Maximum willingness to pay
- π: Fee Priority Multiplier
- R: Read Set
- W: Write Set

Objects and Transactions



Object: Data that any transaction reads, modifies, or creates.

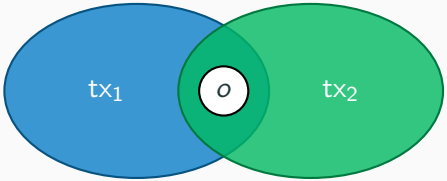
$$\mathsf{tx} := (t, g, \pi, \{R, W\})$$

- t: Compute time (upper bound)
- g: Maximum willingness to pay
- ~~π: Fee Priority Multiplier~~
- ~~R: Read Set~~
- W: Write Set

Conflicts and Schedules

Conflicts

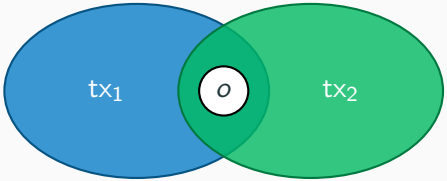
Two transactions conflict if they share an object (and one of them writes it).



Conflicts and Schedules

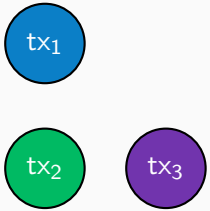
Conflicts

Two transactions conflict if they share an object (and one of them writes it).

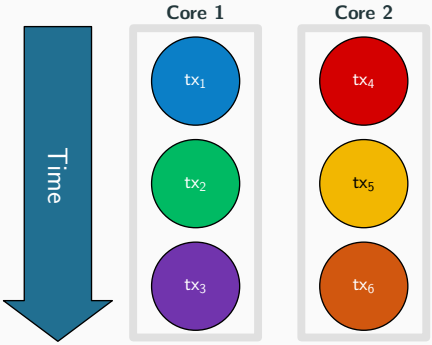


Schedules

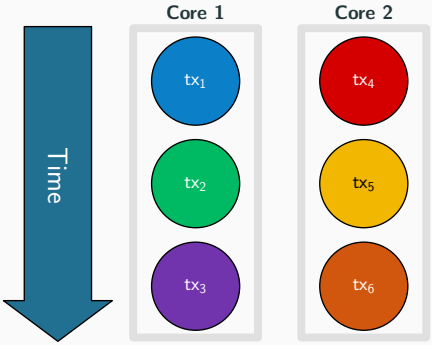
To avoid conflicts from being executed in parallel, a scheduler generates a non-overlapping schedule for execution.



Scheduler's Objective

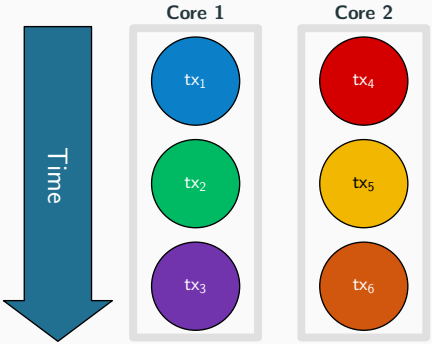


Scheduler's Objective



Maximize (weighted by fee) the number of transactions scheduled

Scheduler's Objective



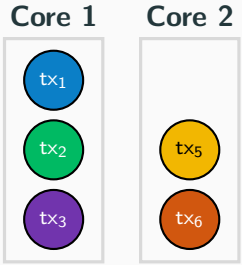
Maximize (weighted by fee) the number of transactions scheduled

Previous Work: Minimize the time taken to execute all transactions!

First Peril:
Shill Transactions

Shill Transactions

Previous work minimizes the time taken to execute all transactions.

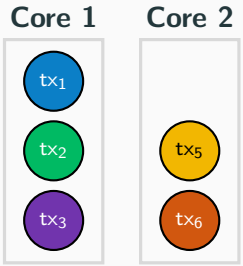


Fee paid by tx_1 is high

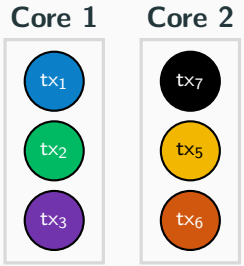
Shill Transactions

Previous work minimizes the time taken to execute all transactions.

Strategy: add a fake transaction.



Fee paid by tx_1 is high



Fee paid by $tx_1 + tx_7$ is lower

User Shill on Shapley GCM

Measures a transaction's contribution to makespan using its Shapley value.

Setup
$T = \{\mathsf{tx}_i = \{1, o_i\}\}$, user's transaction $\mathsf{tx}_u = \{2, o_0\}$. Number of cores $= 14$.

Without Shill

$$\mathrm{gas}_T^{\mathrm{Shap}}(\mathsf{tx}_u) = \frac{15}{14} \approx 1.07$$

User Shill on Shapley GCM

Measures a transaction's contribution to makespan using its Shapley value.

Setup

$T = \{\mathsf{tx}_i = \{1, o_i\}\}$, user's transaction $\mathsf{tx}_u = \{2, o_0\}$. Number of cores $= 14$.

Without Shill

$$\mathrm{gas}_T^{\mathrm{Shap}}(\mathsf{tx}_u) = \frac{15}{14} \approx 1.07$$

With Shill

Add $\mathsf{tx}_{s_1} \simeq (1, \{o_1\})$, $\mathsf{tx}_{s_2} \simeq (1, \{o_2\})$.

$$\mathrm{gas}_T^{\mathrm{Shap}}(\mathsf{tx}_u) = \underbrace{\frac{143}{240}}_{\mathsf{tx}_u} + 2 \cdot \underbrace{\frac{43}{240}}_{\text{shills}} \approx 0.95$$

User Shill on Shapley GCM

Measures a transaction's contribution to makespan using its Shapley value.

Setup

$T = \{\mathsf{tx}_i = \{1, o_i\}\}$, user's transaction $\mathsf{tx}_u = \{2, o_0\}$. Number of cores $= 14$.

Without Shill

$$\mathrm{gas}_T^{\mathrm{Shap}}(\mathsf{tx}_u) = \frac{15}{14} \approx 1.07$$

With Shill

Add $\mathsf{tx}_{s_1} \simeq (1, \{o_1\})$, $\mathsf{tx}_{s_2} \simeq (1, \{o_2\})$.

$$\mathrm{gas}_T^{\mathrm{Shap}}(\mathsf{tx}_u) = \underbrace{\frac{143}{240}}_{\mathsf{tx}_u} + 2 \cdot \underbrace{\frac{43}{240}}_{\text{shills}} \approx 0.95$$

Attack outcome

The user pays less after adding fake transactions.

User Shill Attacks

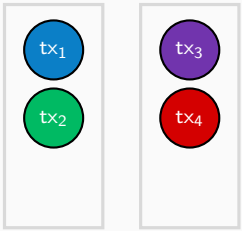
	W. Area	Shapley	Banzhaf	TPM	ESM	XSM [1]
User shill		×	×			×

$\times$ = vulnerable to shill attacks

[1] All mechanisms proposed in Acilan et al.

Shill Transactions

Previous work minimizes the time taken to execute all transactions.

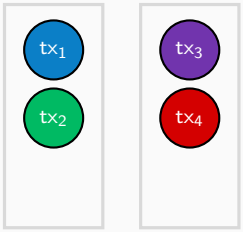


Fee paid by tx_1 is low

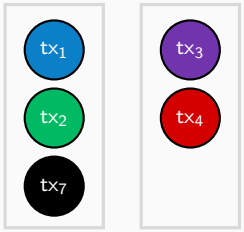
Shill Transactions

Previous work minimizes the time taken to execute all transactions.

Scheduler strategy: add a fake transaction.



Fee paid by tx_1 is low



Fee paid by tx_1 increases

Assuming no burned fee.

Scheduler Shill on TPM GCM

Time-proportional share of total makespan.

Setup
$\mathsf{tx}_1 \simeq (6, \{o_1\})$, $\mathsf{tx}_2 \simeq (6, \{o_2\})$.

Without Shill

$$\mathrm{gas}_T^{\mathrm{TPM}}(\mathsf{tx}_1) = \frac{6}{12} \cdot 6 = 3$$

$$\text{Revenue} = 3 + 3 = 6.$$

With Shill

Add $\mathsf{tx}_3 \simeq (6, \{o_1, o_2\})$:

$$\mathrm{gas}_T^{\mathrm{TPM}}(\mathsf{tx}_1) = \frac{6}{18} \cdot 12 = 4$$

$$\text{Revenue} = 4 + 4 = 8.$$

Attack outcome
The scheduler earns more after adding a fake transaction.

Scheduler Shill Attacks

	W. Area	Shapley	Banzhaf	TPM	ESM	XSM [1]
Scheduler shill		×	×	×	×	×

$\times$ = vulnerable to shill attacks

Assuming no burned fee.

[1] All mechanisms proposed in Acilan et al.

Second Peril:
Uncertain Execution

Contingency of Transactions

AMM Swap Example

Read price (o_1)

if price $>$ threshold:

- Modify balance (o_2)
- Update liquidity (o_3)

else: o_2, o_3 go unused.

Contingency of Transactions

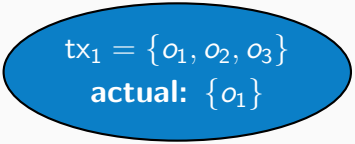
AMM Swap Example

Read price (o_1)

if price $>$ threshold:

- Modify balance (o_2)
- Update liquidity (o_3)

else: o_2, o_3 go unused.



Contingency of Transactions

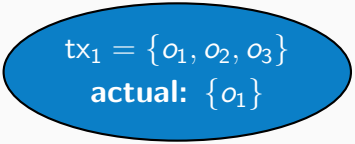
AMM Swap Example

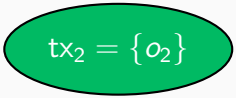
Read price (o_1)

if price $>$ threshold:

- Modify balance (o_2)
- Update liquidity (o_3)

else: o_2, o_3 go unused.





Still a conflict at scheduling time!

Contingency of Transactions

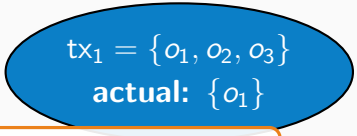
AMM Swap Example

Read price (o_1)

if price > threshold:

- Modify o_2
- Update liquidity (o_3)

else: o_2, o_3 go unused.



Should the user pay for unused objects or compute?

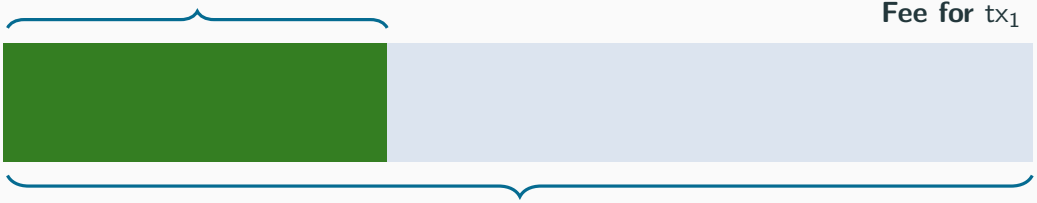
Still a conflict at scheduling time!

User and Scheduler Risk



User and Scheduler Risk

Fee if only charged for used objects



Fee if charged for all declared objects

User and Scheduler Risk

Fee if only charged for used objects



Fee if charged for all declared objects

User and Scheduler Risk



User and Scheduler Risk



User Risk: Charge user pays for unused resources

User and Scheduler Risk



User Risk: Charge user pays for unused resources

Scheduler Risk: Loss in revenue for the scheduler due to unused resources.

Contingency Makes Shills Cheaper

If users pay only for what they actually use, shills can deliberately under-execute, and pay close to nothing.

	W. Area	Shapley	Banzhaf	TPM	ESM	XSM
User shill	$\times^{lr}$	$\times$	$\times$	$\times$	$\times$	$\times$
Scheduler shill	$\times^{lr}$	$\times$	$\times$	$\times$	$\times$	$\times$

Contingency Makes Shills Cheaper

If users pay only for what they actually use, shills can deliberately under-execute, and pay close to nothing.

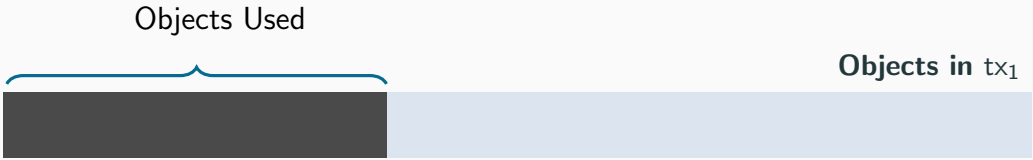
	W. Area	Shapley	Banzhaf	TPM	ESM	XSM
User shill	$\times^{lr}$	$\times$	$\times$	$\times$	$\times$	$\times$
Scheduler shill	$\times^{lr}$	$\times$	$\times$	$\times$	$\times$	$\times$

lr: Long-range attack needed.

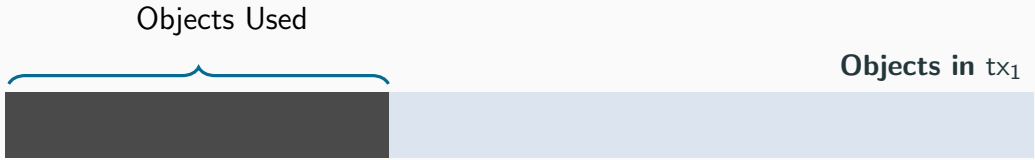
Determining Fee

Objects in tx_1

Determining Fee

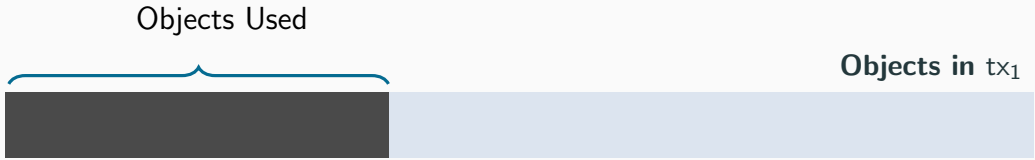


Determining Fee



User pays only for what it uses,
e.g., Ethereum.

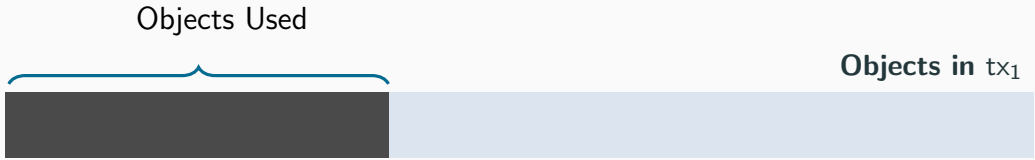
Determining Fee



User pays only for what it uses,
e.g., Ethereum.

User pays for all declared resources,
e.g., Monad.

Determining Fee

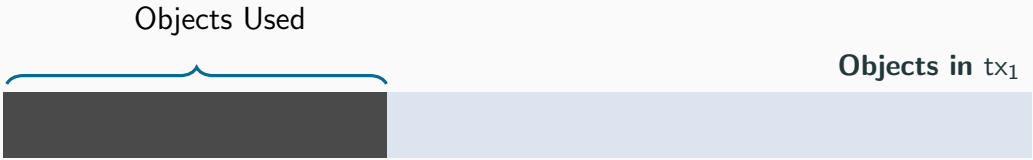


User pays only for what it uses,
e.g., Ethereum.

User pays for all declared resources,
e.g., Monad.

Somewhere in between?

Determining Fee



User pays only for what it uses,
e.g., Ethereum.

User pays for all declared resources,
e.g., Mina.

Somewhere in between?

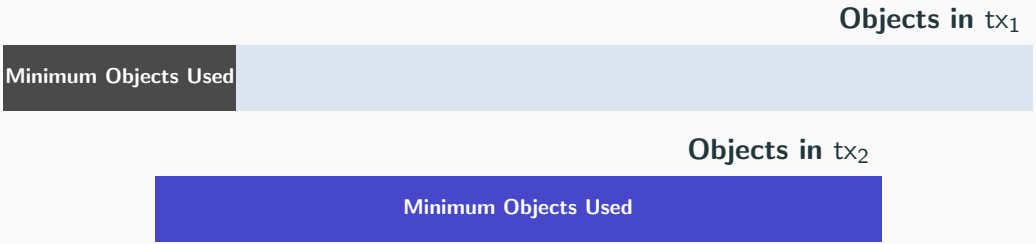
And how much of the paid fee should actually be given to the scheduler?

Determining Schedule

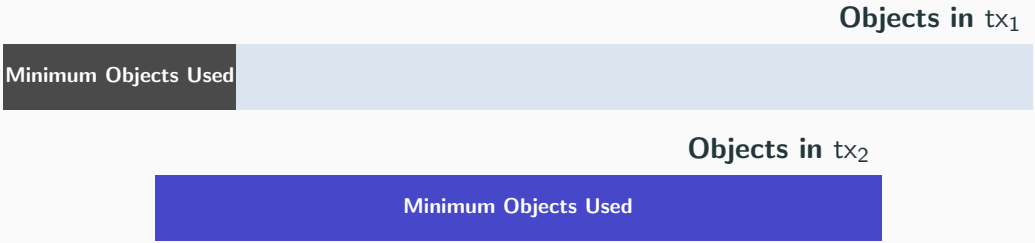
Objects in tx_1

Minimum Objects Used	

Determining Schedule



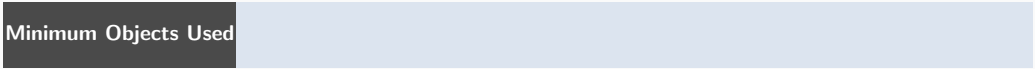
Determining Schedule



Scheduler's choice: **Optimistic**
Chooses tx_1

Determining Schedule

Objects in tx_1



Objects in tx_2



Scheduler's choice: **Optimistic**
Chooses tx_1

Scheduler's choice: **Pessimistic**
Chooses tx_2

Results and Takeaways

OW-TFM: price the objects

Every object o carries its own price $p(o)$, updated block to block from how much o got used — an EIP-1559-style rule, per object.

OW-TFM: price the objects

Every object o carries its own price $p(o)$, updated block to block from how much o got used — an EIP-1559-style rule, per object.

A transaction then pays,

$$f(\mathrm{tx}) \;=\; \pi \cdot \left(\sum_{o \,\in\, \mathrm{used}(\mathrm{tx})} p(o) \cdot t \;+\; \alpha \cdot \sum_{o \,\in\, \mathrm{unused}(\mathrm{tx})} p(o) \cdot t \right)$$

OW-TFM: Preventing long-range shills

Shills don't vanish — they get expensive

- **In-block:** prices are already fixed by history. A fake transaction moves nothing.
- **Across blocks:** to move $p(o)$ you can only increase the usage of o in previous blocks. (Only scheduler based shill attacks possible!)

$\Rightarrow$ cost incurred per block: $(1-\gamma)\,f$

(Since only γ fraction of fee is retained!)

OW-TFM: Preventing long-range shills

Shills don't vanish — they get expensive

- **In-block:** prices are already fixed by history. A fake transaction moves nothing.
- **Across blocks:** to move $p(o)$ you can only increase the usage of o in previous blocks. (Only scheduler based shill attacks possible!)

$\Rightarrow$ cost incurred per block: $(1-\gamma)\,f$

(Since only γ fraction of fee is retained!)

As long as the cost incurred is larger than theoretical maximum revenue increase, OW-TFM is safe against shill attacks!

Summary

1. **A formal model** for parallel execution and contingent resource use.

Summary

1. **A formal model** for parallel execution and contingent resource use.

2. **An impossibility:** at least one of the user or scheduler must bear the cost of contingency. (or all fee is burnt)

Summary

1. **A formal model** for parallel execution and contingent resource use.

2. **An impossibility:** at least one of the user or scheduler must bear the cost of contingency. (or all fee is burnt)

3. **OW-TFM:** price each object separately.
   Parameterizations required for shill-proofness

Future Research Directions

1. **The scheduling algorithm:** We sidestep by only analyzing per-object pricing based on history.

Future Research Directions

1. **The scheduling algorithm:** We sidestep by only analyzing per-object pricing based on history.

2. We avoid **Concurrency Control** by scheduling no conflicts: Using concurrency control methods for online re-scheduling?

Future Research Directions

1. **The scheduling algorithm:** We sidestep by only analyzing per-object pricing based on history.

2. We avoid **Concurrency Control** by scheduling no conflicts: Using concurrency control methods for online re-scheduling?

3. Pricing efficient object usage: In conflict with parallelization!

Perils of Parallelism

Perils of Parallelism: Transaction Fee Mechanisms under Execution Uncertainty

Sarisht Wadhwa* *Duke University*

Aviv Yaish* *Yale University, IC3, Complexity Science Hub Vienna*

Fan Zhang *Yale University, IC3*

Kartik Nayak *Duke University*



Contact: wsarisht@gmail.com

Link to paper