Fairtopia： A Democratized Cloud-hosted Financial Exchange Platform

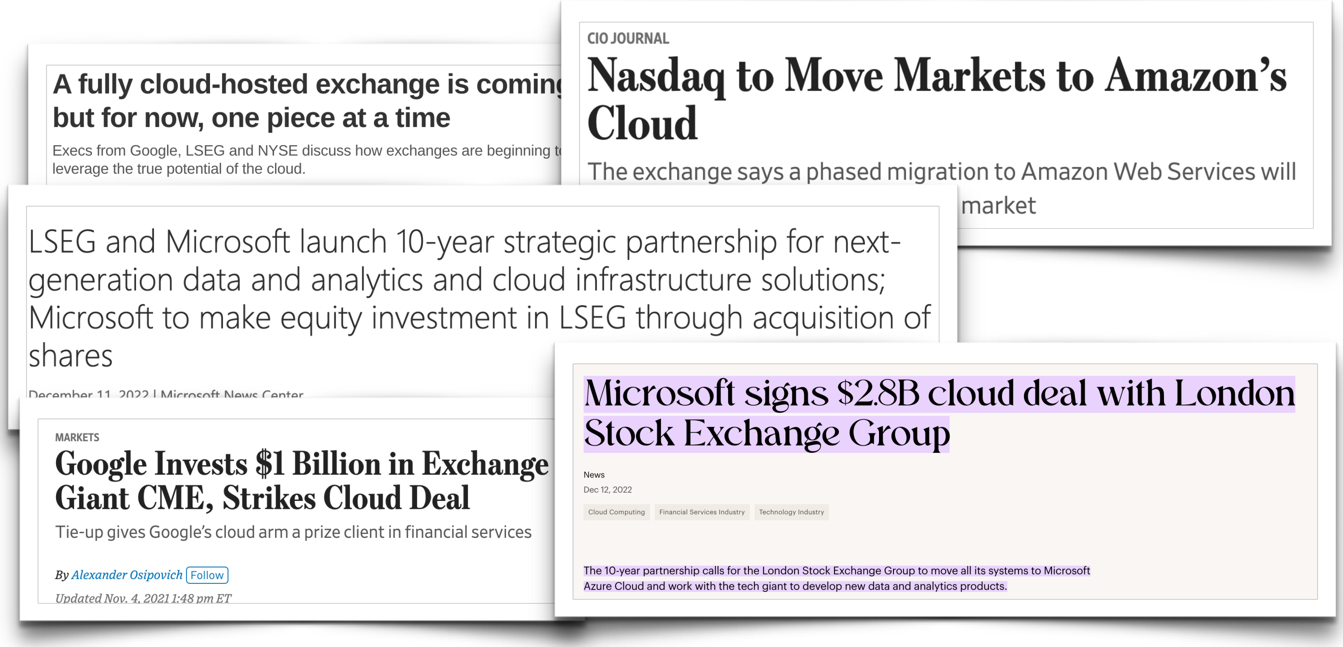
—— Pushing Fairness To Extreme via Communication and Computation Synchrony

***Presenter**: Liangcheng (LC) Yu*

***Mentors**: Prateesh Goyal, Ilias Marinos*

***Date:** October 9, 2023*

Rising interest in cloud-hosted exchange services



A fully cloud-hosted exchange is comin
but for now, one piece at a time

Execs from Google, LSEG and NYSE discuss how exchanges are beginning t
leverage the true potential of the cloud.

CIO JOURNAL

Nasdaq to Move Markets to Amazon's Cloud

The exchange says a phased migration to Amazon Web Services will
market

LSEG and Microsoft launch 10-year strategic partnership for next-generation data and analytics and cloud infrastructure solutions; Microsoft to make equity investment in LSEG through acquisition of shares

December 11, 2022 | Microsoft News Center

MARKETS

Google Invests $1 Billion in Exchange Giant CME, Strikes Cloud Deal

Tie-up gives Google's cloud arm a prize client in financial services

By *Alexander Osipovich* Follow

Updated Nov. 4, 2021 1:48 pm ET

Microsoft signs $2.8B cloud deal with London Stock Exchange Group

News

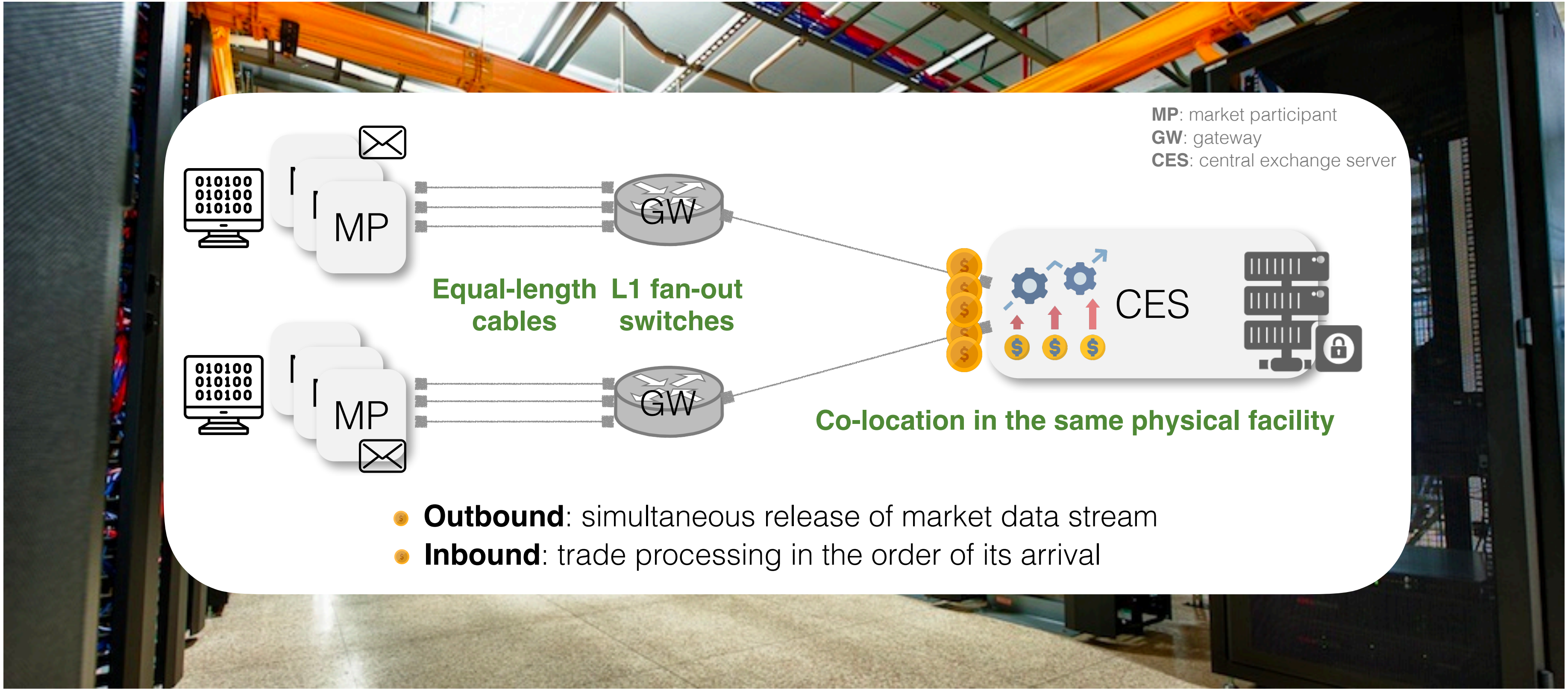
Dec 12, 2022

Cloud Computing | Financial Services Industry | Technology Industry

The 10-year partnership calls for the London Stock Exchange Group to move all its systems to Microsoft Azure Cloud and work with the tech giant to develop new data and analytics products.

- System scalability and resource elasticity
- Cost reduction and ease of management
- Rise of remote work
- …

Fairness, in on-premise infrastructure



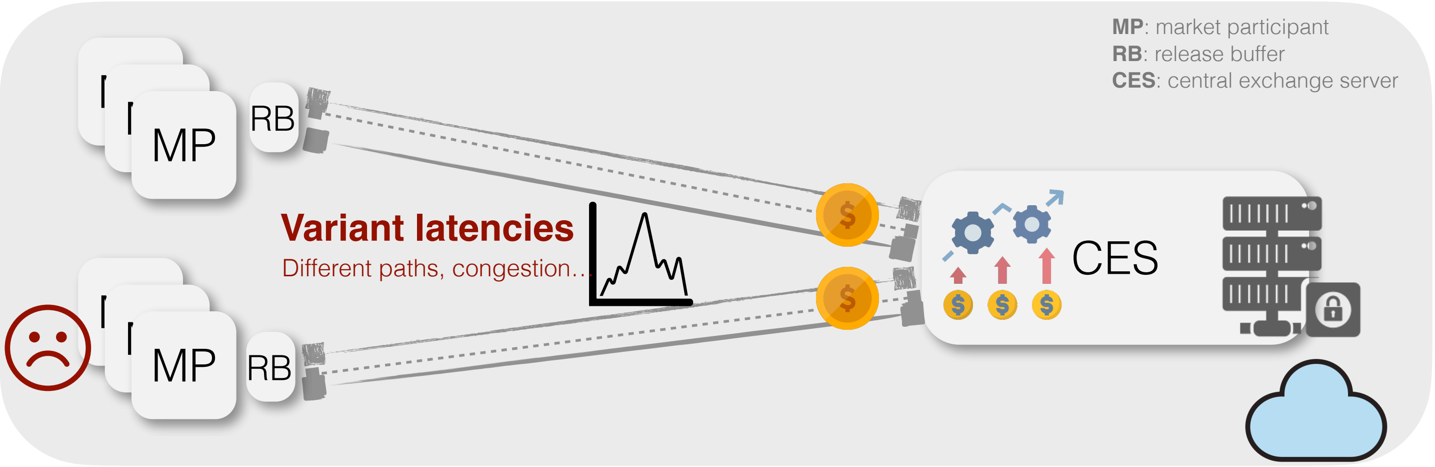
Equal-length cables

L1 fan-out switches

Co-location in the same physical facility

- **Outbound**: simultaneous release of market data stream
- **Inbound**: trade processing in the order of its arrival

Fairness, in the cloud

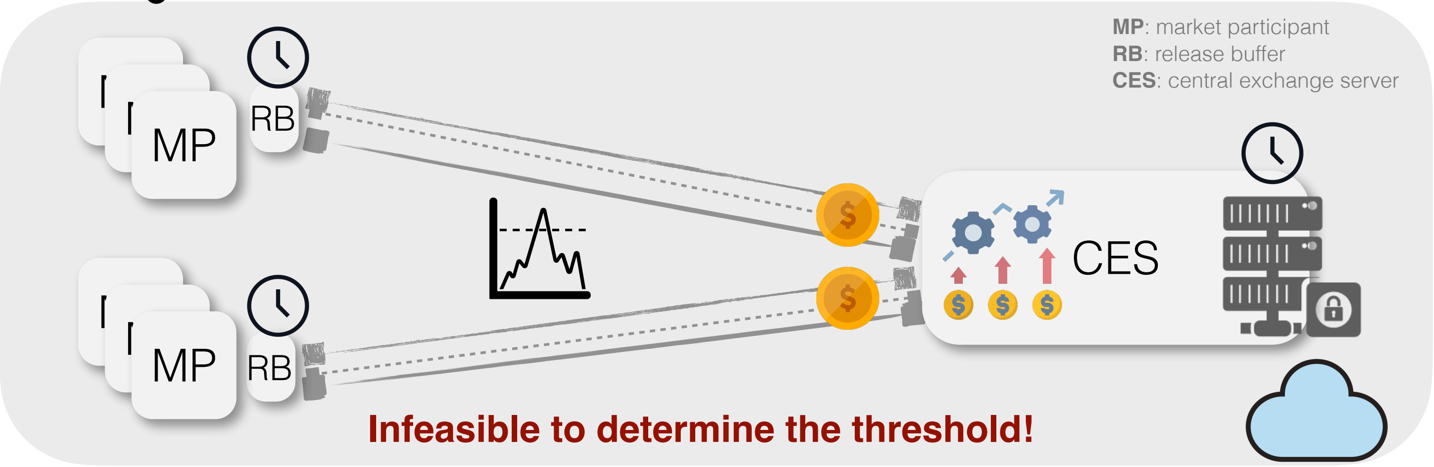


- **Outbound**: simultaneous release of ~~market data stream~~
- **Inbound**: ~~trade processing~~ in the order of its arrival

Unfairness!

Fairness, in CloudEx〔HotOS ’21〕

💡 **Idea**: clock synchronization + message inhibition



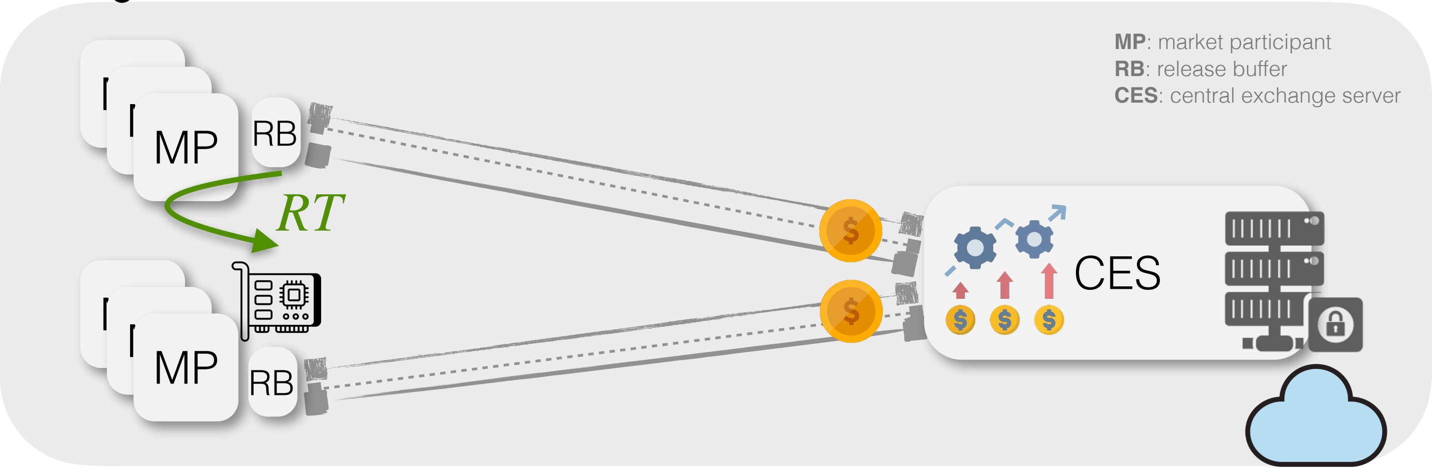
Infeasible to determine the threshold!

😐 Perfect clock synchronization is **hard**

☹ Latencies are **unpredictable** and **unbounded**

Fairness, in DBO〔SIGCOMM ’23, HotNets ’22〕

💡 **Idea**: logical delivery clock based on **response time (RT)**

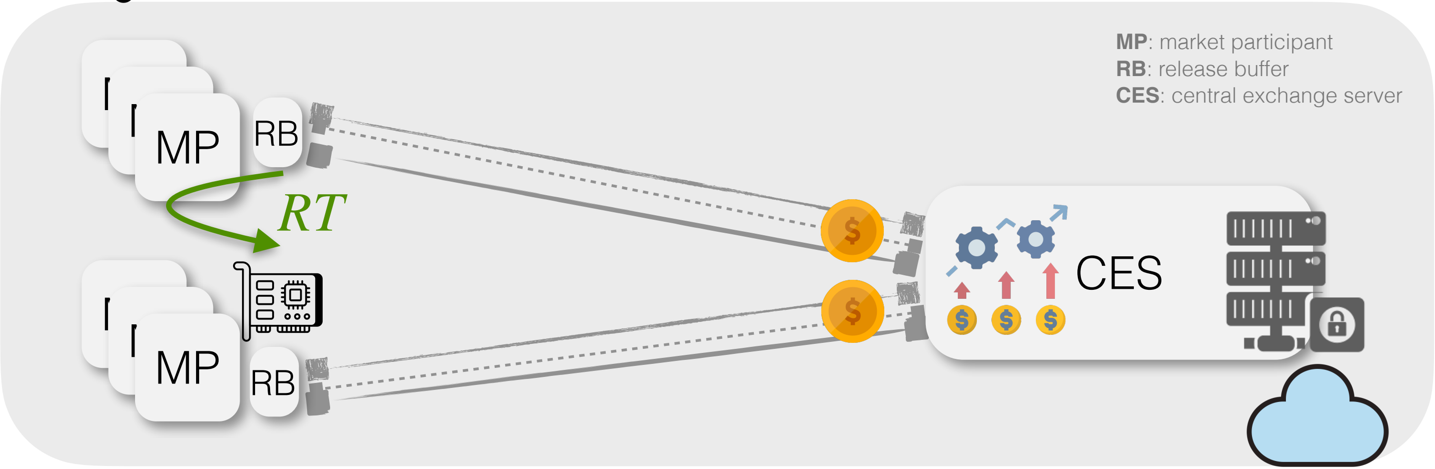


☺ Relaxation of clock sync. requirement via post-hoc correction

☺ Guaranteed fairness (tailored to ***trigger-point based, high-speed trades***)

Fairness, in DBO [SIGCOMM '23, HotNets '22]

Idea: logical delivery clock based on **response time (RT)**



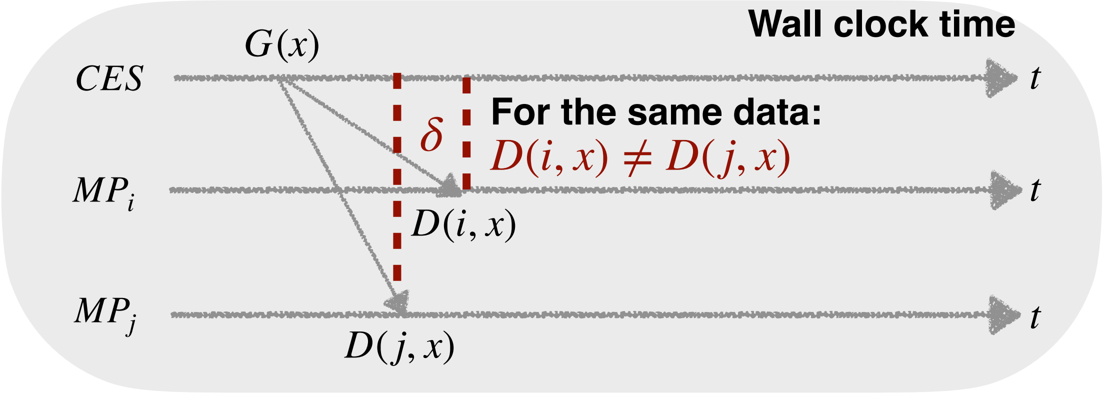
MP: market participant
RB: release buffer
CES: central exchange server

Relaxation of clock sync. requirement via post-hoc correction

Can we go beyond the existing paradigm and extend the fairness notion?

Guaranteed fairness (tailored to trigger-point based, high-speed trades)

Let's reflect on underlying model today…



Wall clock time

For the same data:
$D(i,x) \neq D(j,x)$

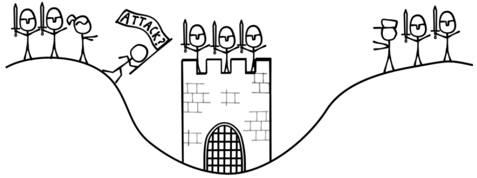
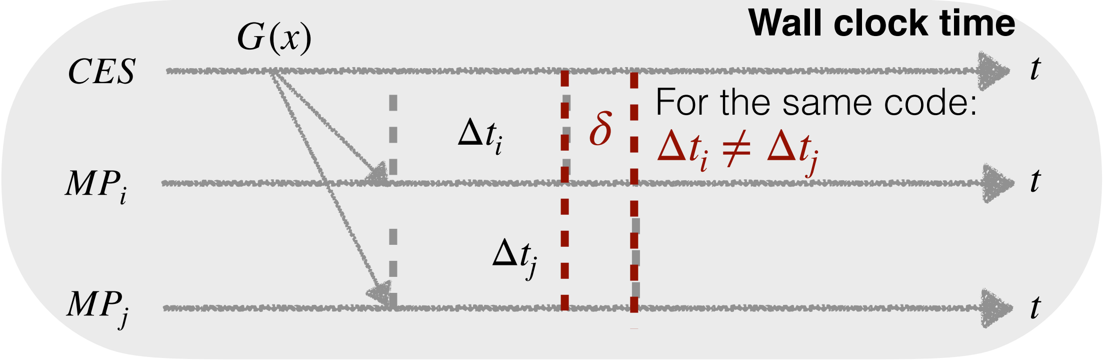


Image source: https://haydenjames.io/the-two-generals-problem/

Truly simultaneous delivery is ***impossible***!

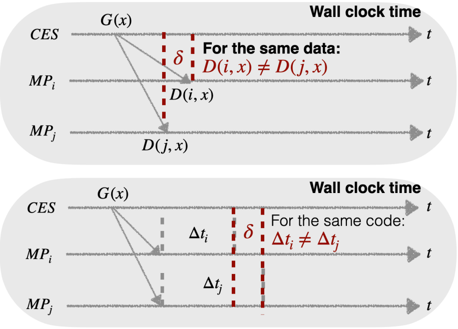


Wall clock time

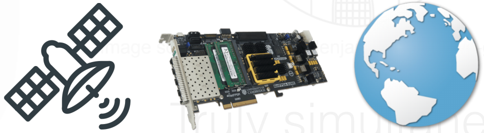
For the same code:
$\Delta t_i \neq \Delta t_j$

Computation can be ***non-deterministic*** at $O(\mu s)$
(thermal condition, resource utilization…)

Let’s reflect on underlying model today…



Communication and computation synchrony are ***challenging*** in real-time



Symptoms: trading ***arms race*** to gain δ advantage and increasing market ***consolidation***

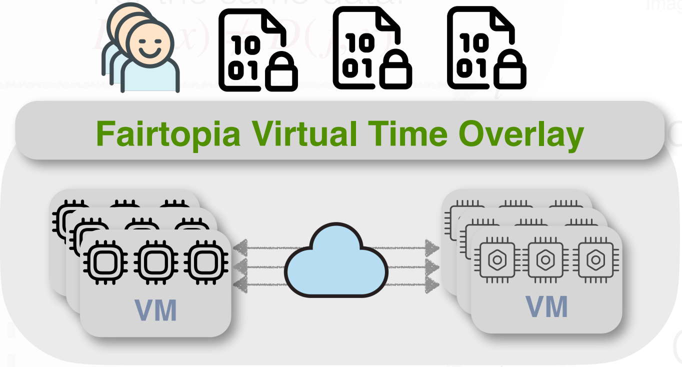
Truly simultaneous delivery is ***impossible***!

Computation can be ***non-deterministic*** at $O(\mu s)$ (thermal condition, resource utilization…)

Synchrony is challenging in real time

Can we **guarantee fairness** via achieving **communication** and **computation synchrony**?

Fairtopia: A Democratized Financial Exchange Platform



- Determinism w.r.t. underlying communication & computation
- Generality to trading patterns
- Democratized competition for special hardware

For the same data:

Image source: https://haydenjames.io/the-two-generals-problem/

Truly simultaneous delivery is *impossible*!

For the same code:

Computation can be *non-deterministic* at $O(\mu s)$ (thermal condition, resource utilization…)

Fairtopia outline

- Conceptual foundation
- User abstraction
- Demo of the real system

This presentation

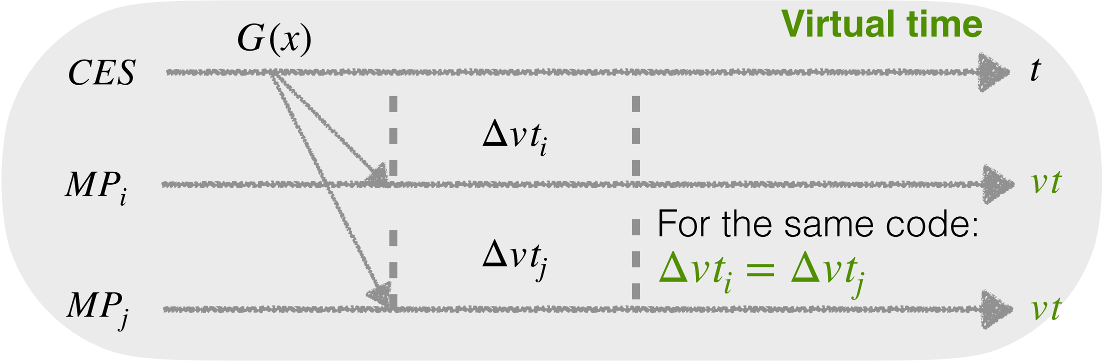
- Implementation and benchmarks

Fairtopia outline

- **Conceptual foundation**
- User abstraction
- Demo of the real system
- Implementation and benchmarks

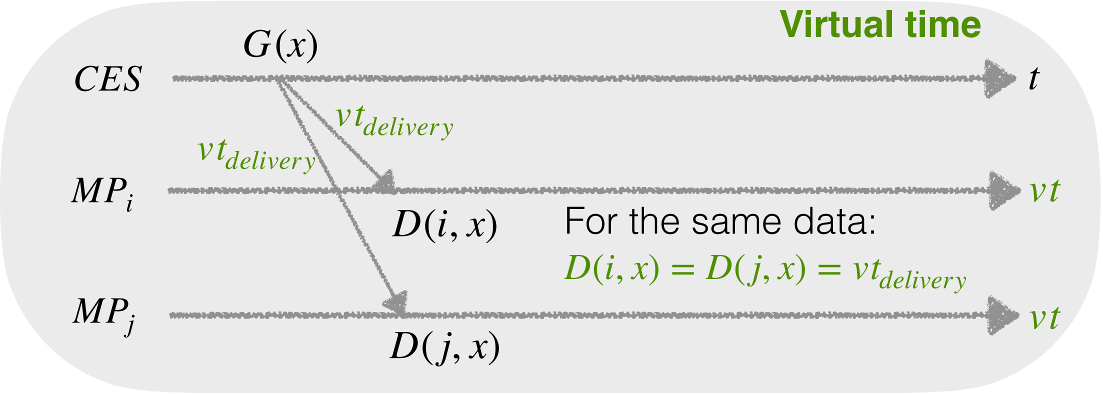
Impossible? Imagine in virtual time domain ···

Virtual time unit ≡ *some equal amount of work*



For the same code:
$\Delta vt_i = \Delta vt_j$

Quantizing vt per ***'actual amount of work'*** for computation synchrony



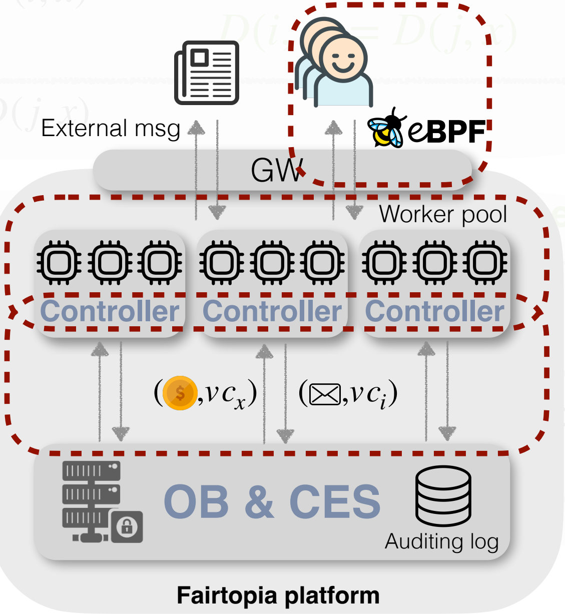
For the same data:
$D(i, x) = D(j, x) = vt_{delivery}$

Freezing and ***advancing*** vt for communication synchrony

How to implement a real system?

Instantiate vt as virtual cycles of a platform-agnostic IR/VM

Account and **control** the advancement of virtual cycles



Fairtopia platform

1 **Programming interface**

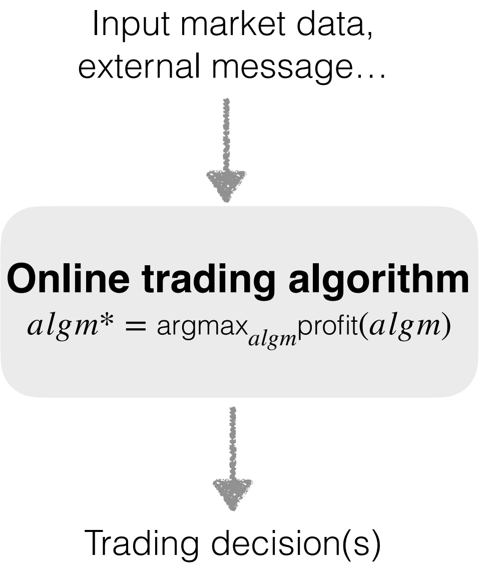
3 **Runtime execution**

2 **Virtual cycle tracking**

Fairtopia outline

- Conceptual foundation
- **User abstraction**
- Demo of the real system
- Implementation and benchmarks

User programming abstraction



Input market data, external message…

Online trading algorithm

$algm^* = \text{argmax}_{algm}\text{profit}(algm)$

Trading decision(s)

```
#include <fairtopia_user.h>

int mu_handler(subscribed_context_t* data) {
    if ((*data) > 100) {
        // Sell
        trade_t trade = 1;
        submit_trade(&trade);
    } else if ((*data) < 10) {
        // Buy
        trade_t trade = 2;
        submit_trade(&trade);
    }
    map_update(0, &trade);
    return 0;
}
```

White-list set of extensible service APIs

Just-in-time trade submission

Narrow KV store API (e.g., lookup, update) for stateful invocations

The interface is expressive enough

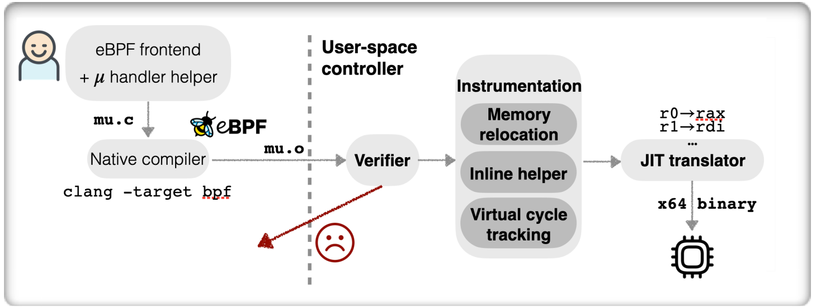
- Fibonacci, Bubble Sort…
- SMA Mean Reversion
- EMA Mean Reversion
- Relative Strength Index
- Moving Average Crossover Strategy
- Bollinger Bands Strategy

- Moving Average Convergence Divergence
- Multiple Moving Average Crossover Strategy
- Parabolic SAR
- On Balance Volume (OBV) + EMA
- Stochastic Oscillator
- Basic Market Making
- …

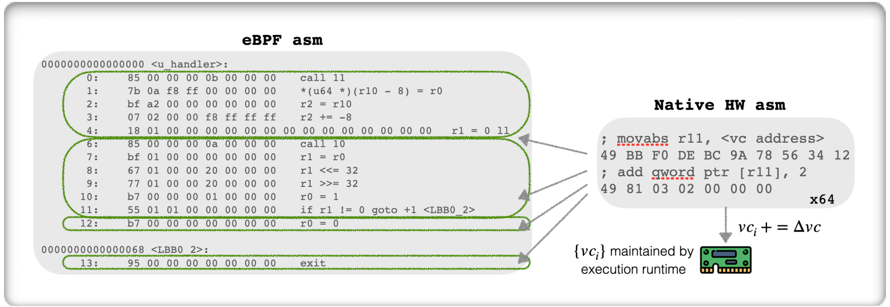
GPT-4

☺ **Running out-of-the-box**

Implementation



Program life time



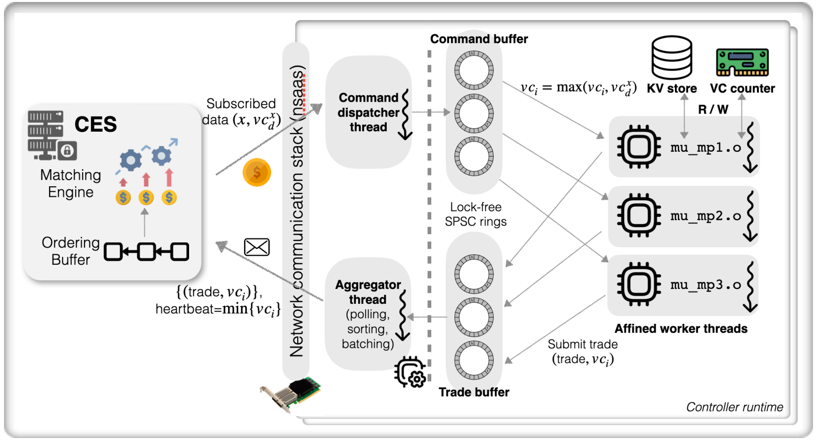
eBPF asm

```
0000000000000000 <u_handler>:
     0:       85 00 00 00 0b 00 00 00    call 11
     1:       7b 0a f8 ff 00 00 00 00    *(u64 *)(r10 - 8) = r0
     2:       bf a2 00 00 00 00 00 00    r2 = r10
     3:       07 02 00 00 f8 ff ff ff    r2 += -8
     4:       18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00    r1 = 0 ll
     6:       85 00 00 00 0a 00 00 00    call 10
     7:       bf 01 00 00 00 00 00 00    r1 = r0
     8:       67 01 00 00 20 00 00 00    r1 <<= 32
     9:       77 01 00 00 20 00 00 00    r1 >>= 32
    10:       b7 00 00 00 01 00 00 00    r0 = 1
    11:       55 01 01 00 00 00 00 00    if r1 != 0 goto +1 <LBB0_2>
    12:       b7 00 00 00 00 00 00 00    r0 = 0

0000000000000068 <LBB0_2>:
    13:       95 00 00 00 00 00 00 00    exit
```

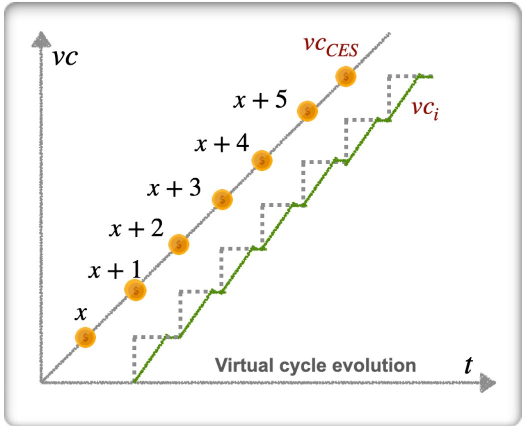
Native HW asm

```
; movabs r11, <vc address>
49 BB F0 DE BC 9A 78 56 34 12
; add qword ptr [r11], 2
49 81 03 02 00 00 00
                         x64
```

Virtual cycle tracking instrumentation

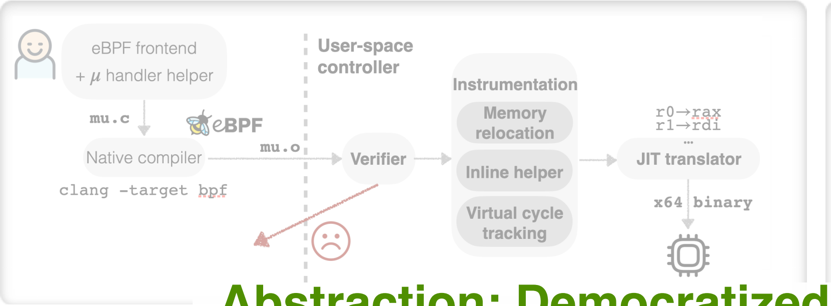

CES
Matching Engine
Ordering Buffer
Subscribed data (x, vc_d^x)
$\{(\text{trade}, vc_i)\}$, heartbeat=$\min\{vc_i\}$
Network communication stack (vsaas)
Command dispatcher thread
Aggregator thread (polling, sorting, batching)
Command buffer
Lock-free SPSC rings
Trade buffer
$vc_i = \max(vc_i, vc_d^x)$
KV store
VC counter
R / W
mu_mp1.o
mu_mp2.o
mu_mp3.o
Affined worker threads
Submit trade (trade, vc_i)
Controller runtime

Runtime execution engine

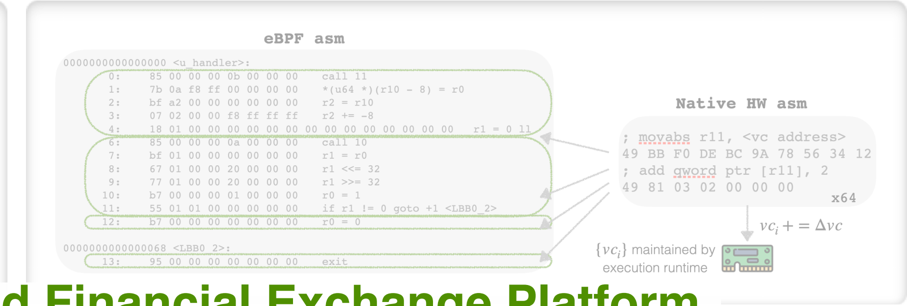


Virtual cycle assignment

Implementation



Program life time



```
0000000000000000 <u_handler>:
   0:    85 00 00 00 0b 00 00 00    call 11
   1:    7b 0a f8 ff 00 00 00 00    *(u64 *)(r10 - 8) = r0
   2:    bf a2 00 00 00 00 00 00    r2 = r10
   3:    07 02 00 00 f8 ff ff ff    r2 += -8
   4:    18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00    r1 = 0 ll
   6:    85 00 00 00 0a 00 00 00    call 10
   7:    bf 01 00 00 00 00 00 00    r1 = r0
   8:    67 01 00 00 20 00 00 00    r1 <<= 32
   9:    77 01 00 00 20 00 00 00    r1 >>= 32
  10:    b7 00 00 00 01 00 00 00    r0 = 1
  11:    55 01 01 00 00 00 00 00    if r1 != 0 goto +1 <LBB0_2>
  12:    b7 00 00 00 00 00 00 00    r0 = 0

0000000000000068 <LBB0_2>:
  13:    95 00 00 00 00 00 00 00    exit
```

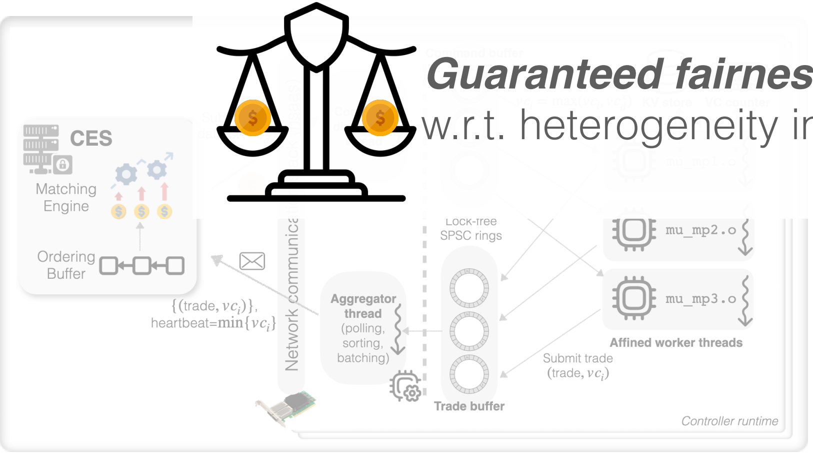
```
; movabs r11, <vc address>
49 BB F0 DE BC 9A 78 56 34 12
; add qword ptr [r11], 2
49 81 03 02 00 00 00
```

Virtual cycle tracking instrumentation

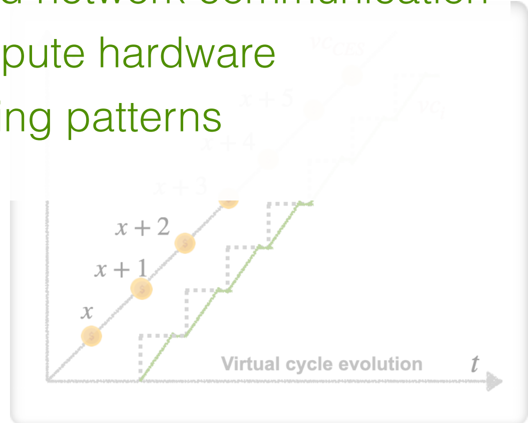
Abstraction: Democratized Financial Exchange Platform

Guaranteed fairness
w.r.t. heterogeneity in:

- Cloud network communication
- Compute hardware
- Trading patterns
- …



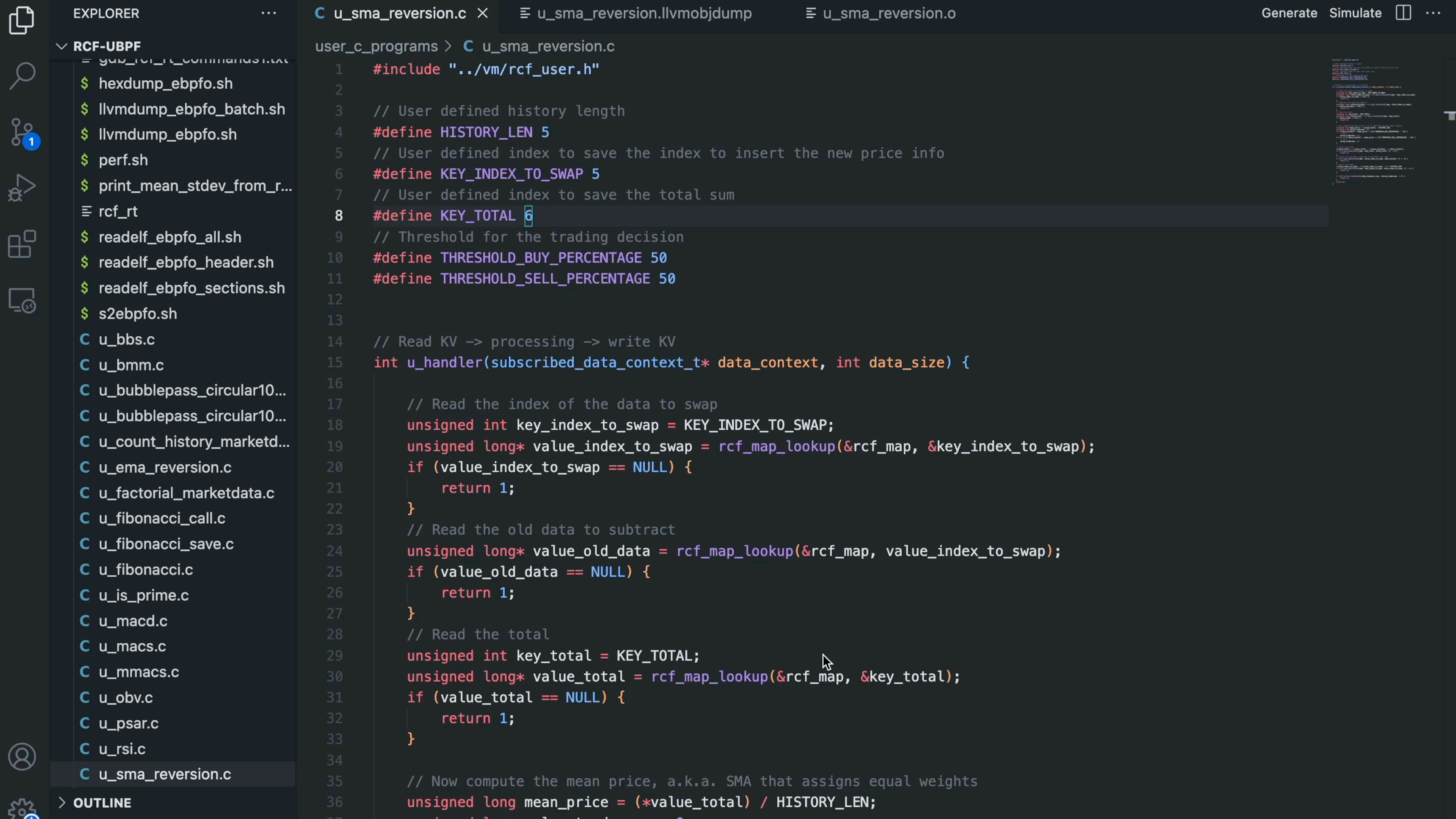
Runtime execution engine



Virtual cycle assignment

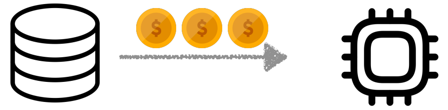
Fairtopia outline

- Conceptual foundation
- User abstraction
- **Demo of the real system**
- Implementation and benchmarks



```c
#include "../vm/rcf_user.h"

// User defined history length
#define HISTORY_LEN 5
// User defined index to save the index to insert the new price info
#define KEY_INDEX_TO_SWAP 5
// User defined index to save the total sum
#define KEY_TOTAL 6
// Threshold for the trading decision
#define THRESHOLD_BUY_PERCENTAGE 50
#define THRESHOLD_SELL_PERCENTAGE 50


// Read KV -> processing -> write KV
int u_handler(subscribed_data_context_t* data_context, int data_size) {

    // Read the index of the data to swap
    unsigned int key_index_to_swap = KEY_INDEX_TO_SWAP;
    unsigned long* value_index_to_swap = rcf_map_lookup(&rcf_map, &key_index_to_swap);
    if (value_index_to_swap == NULL) {
        return 1;
    }
    // Read the old data to subtract
    unsigned long* value_old_data = rcf_map_lookup(&rcf_map, value_index_to_swap);
    if (value_old_data == NULL) {
        return 1;
    }
    // Read the total
    unsigned int key_total = KEY_TOTAL;
    unsigned long* value_total = rcf_map_lookup(&rcf_map, &key_total);
    if (value_total == NULL) {
        return 1;
    }

    // Now compute the mean price, a.k.a. SMA that assigns equal weights
    unsigned long mean_price = (*value_total) / HISTORY_LEN;
```

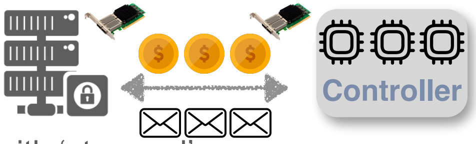
Fairtopia runtime execution is also efficient



Local in-memory market data

Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz

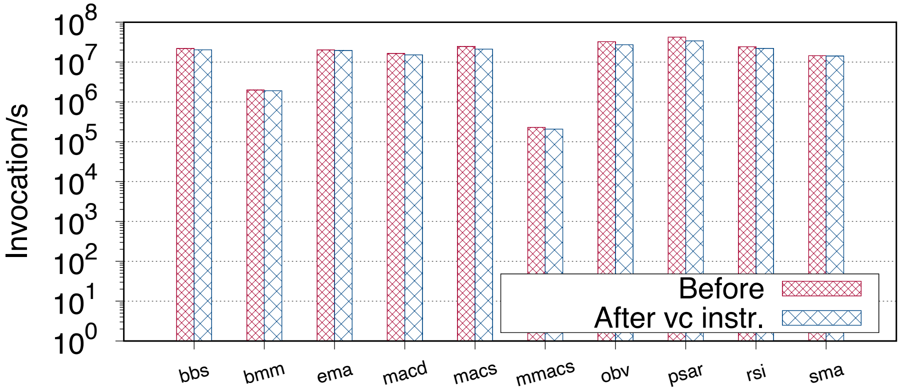
100G ConnectX-6 NIC & Switch port

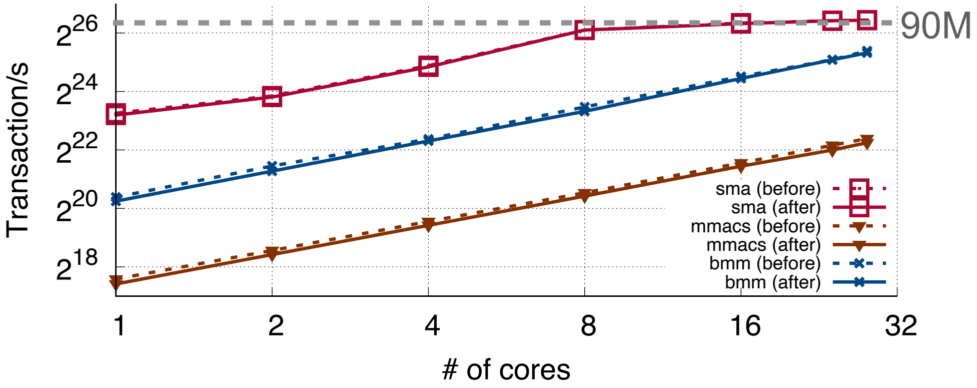


Controller

CES with ‘stressed’ market data + OB

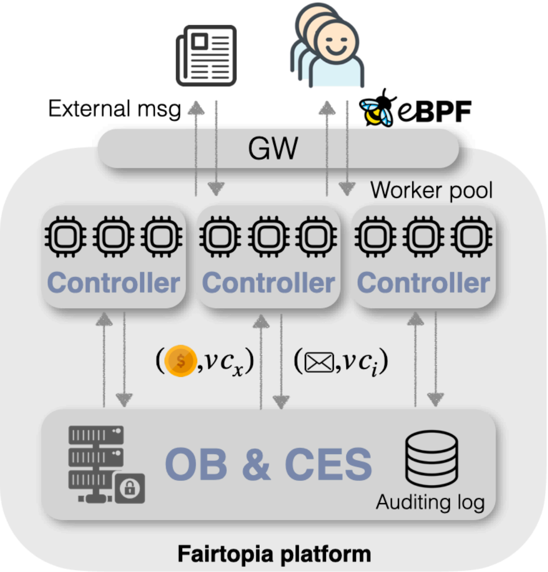
‘Fat’ worker machine





Summary

- Fairtopia: a democratized financial exchange platform
  - Guaranteed fairness with communication and computation synchrony
  - Support of generic trading patterns
  - No need for special hardware
- Opportunities
  - Exploiting determinism: replication/cloud scaling, fault tolerance, and straggler mitigation
  - Extensive testing: virtual cycle assignment protocol w/ latency spikes and external communication
  - Auditing system: prove the history log and dispute the ordering
  - Code privacy and security: host runtime into an enclave



Thank you!