Cuttlefish: A Fair, Predictable Execution Environment for Cloud-hosted Financial Exchange

Liangcheng (LC) Yu, Prateesh Goyal, Ilias Marinos, and Vincent Liu

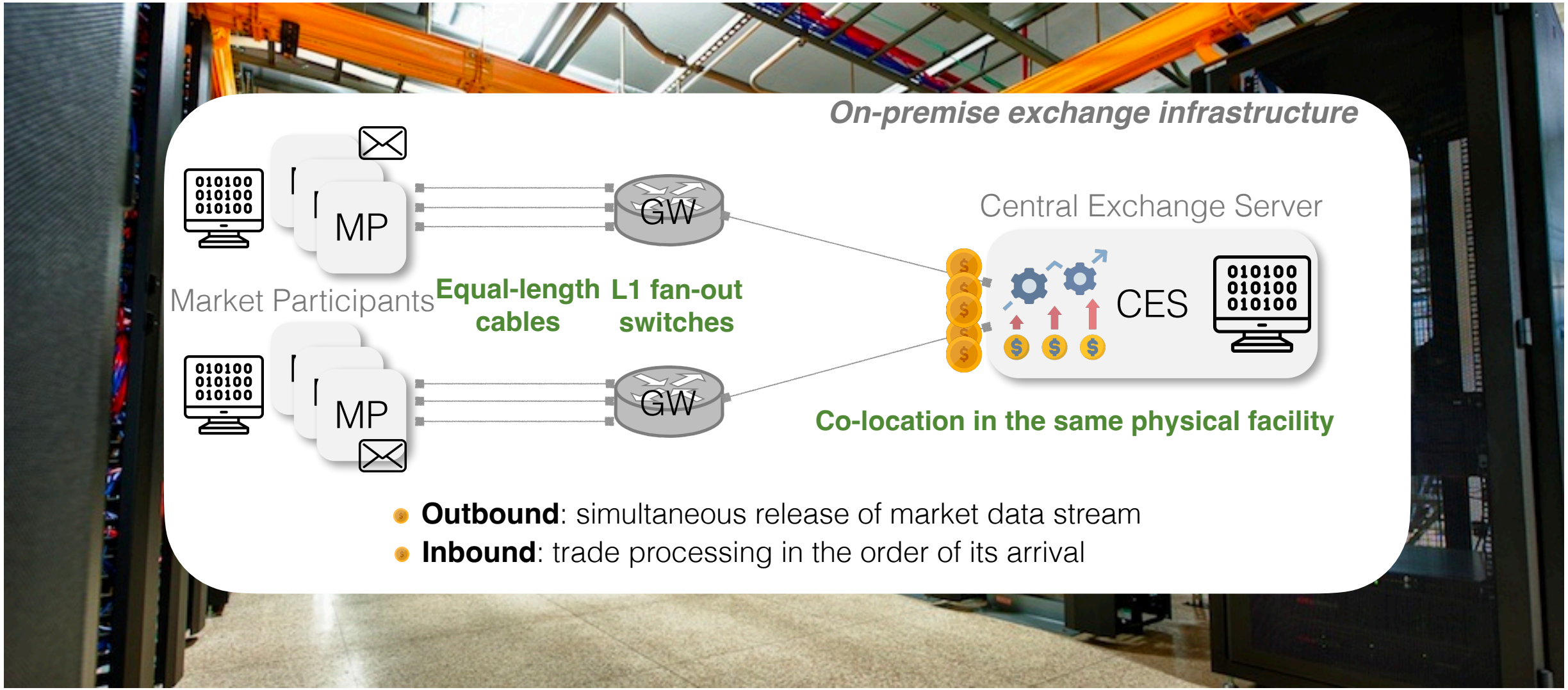
Advances in Financial Technologies (AFT) 2025

Microsoft Research

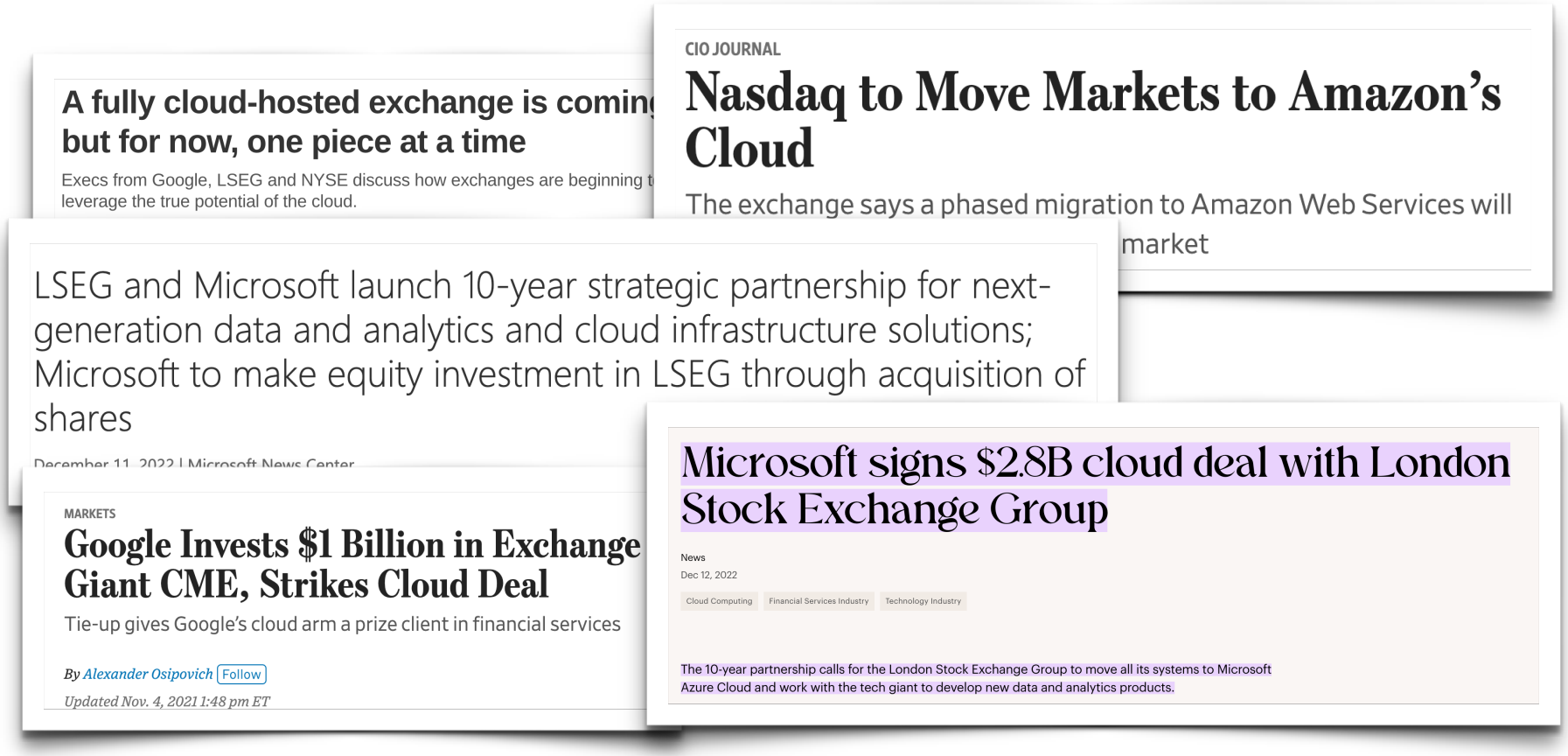
NVIDIA

Penn
University of Pennsylvania

On-premise exchange infrastructure

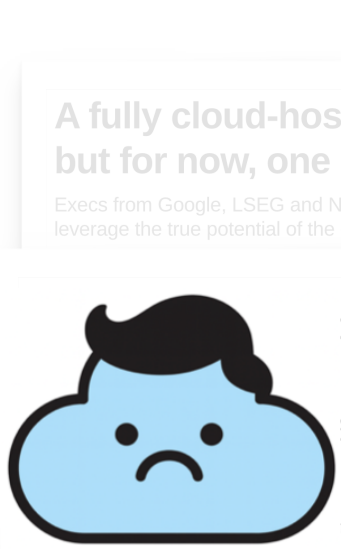


Interest in cloud-hosted exchange services



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

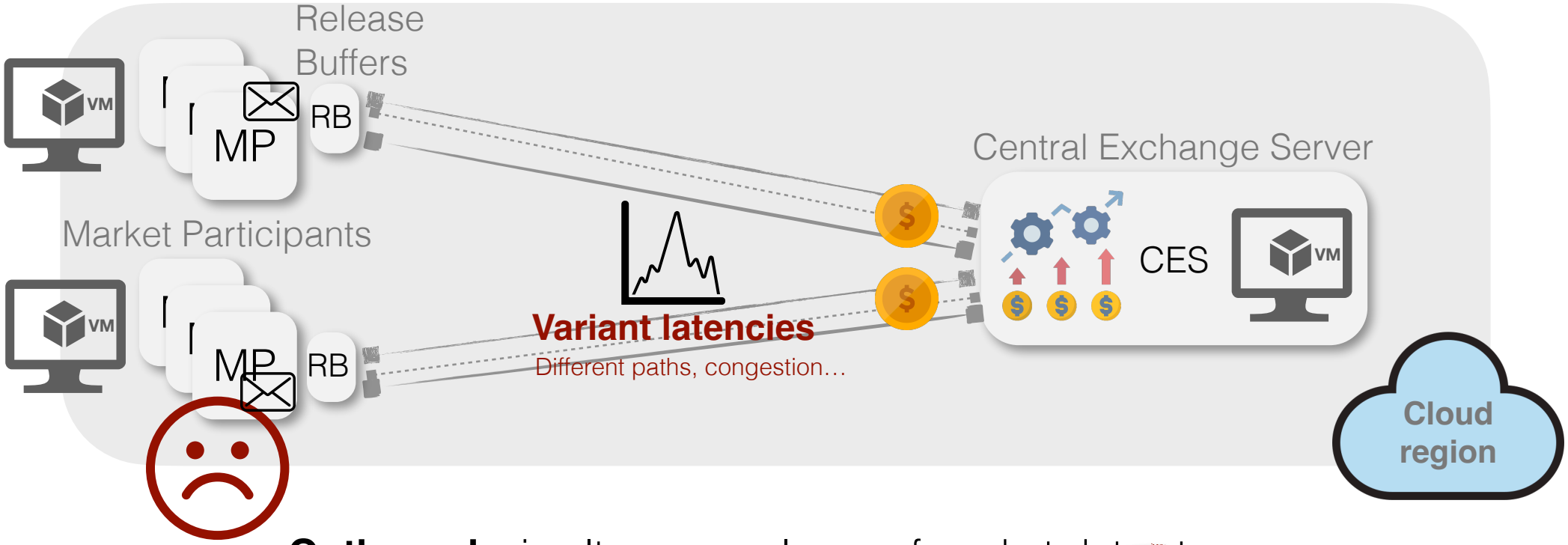
Interest in cloud-hosted exchange services



Cloud infrastructure can introduce unfairness!

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

Variances in network latencies

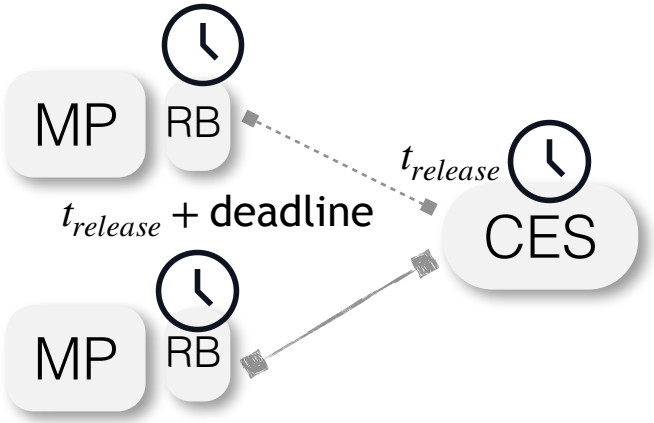


- **Outbound**: simultaneous release of market data stream
- **Inbound**: trade processing in the order of its arrival

Unfairness!

Efforts toward communication fairness

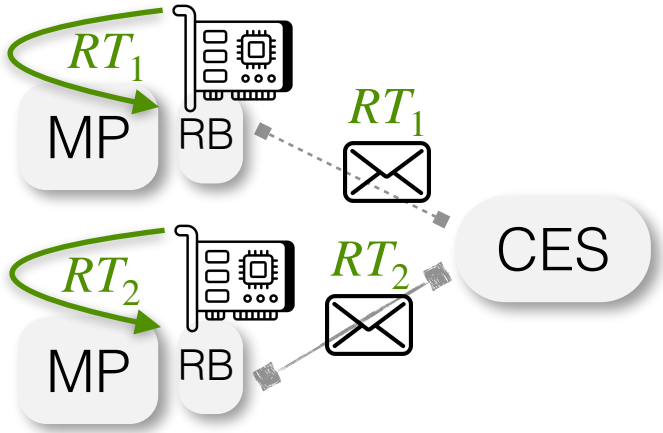
Clock synchronization
(CloudEx, HotOS '21)



😐 Perfect clock synchronization is **hard**

☹ Hard to pre-determine the deadline

Logical clock based on response time (RT)
(DBO, SIGCOMM '23)

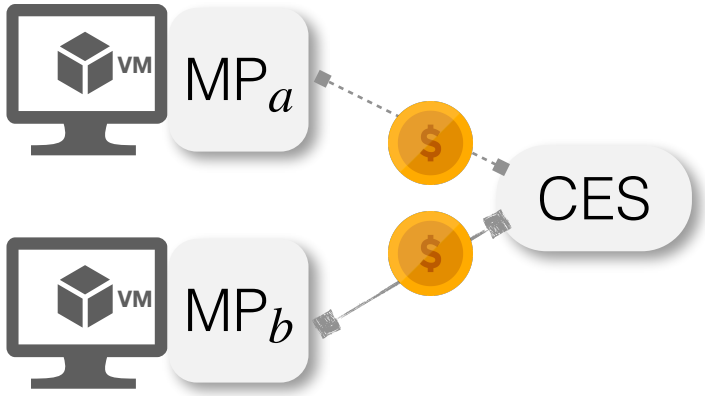


☹ **Limited to trigger-point based trades**

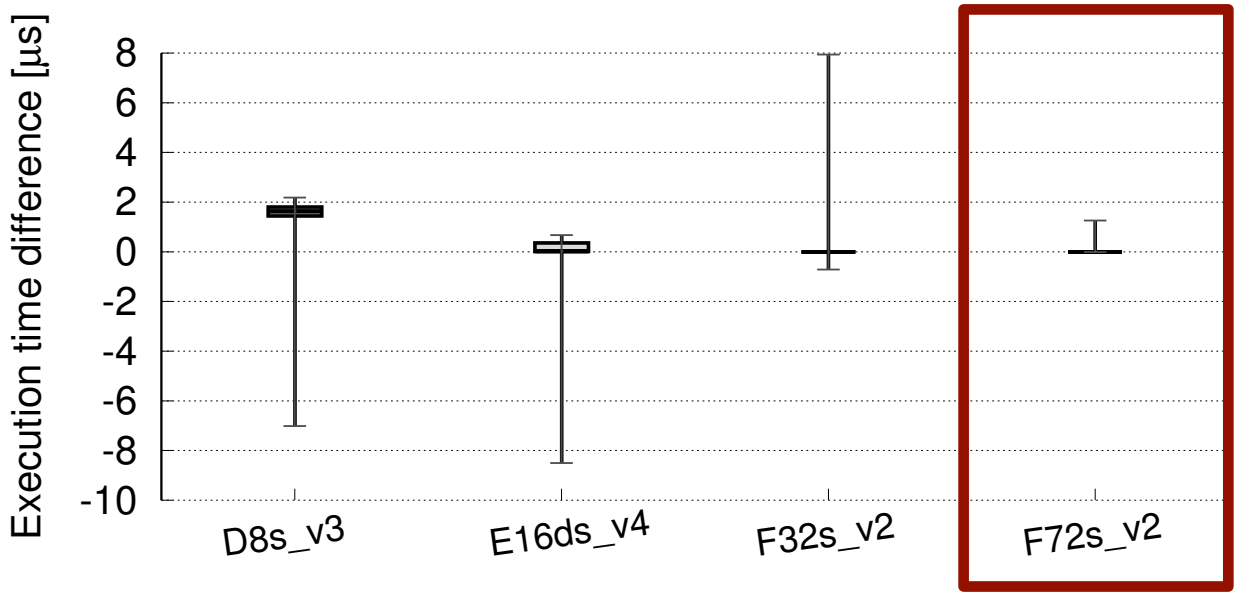
☹ **Doesn't handle MP-RB latency variances**

…cloud execution can also incur unfairness!

- Other sources of unfairness: noisy neighbors, thermal conditions of the processors…

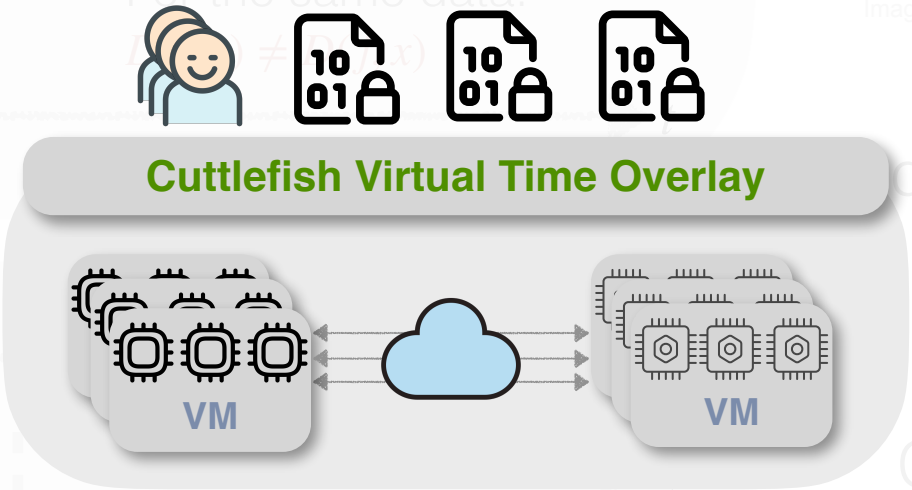


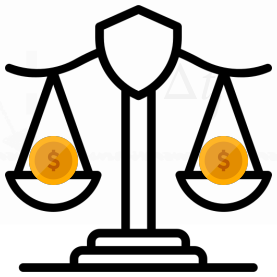
Identical programs running on same types of VMs



Can we **eliminate variations** that come from the cloud infrastructure?

Cuttlefish: A Fair, Predictable Execution Environment





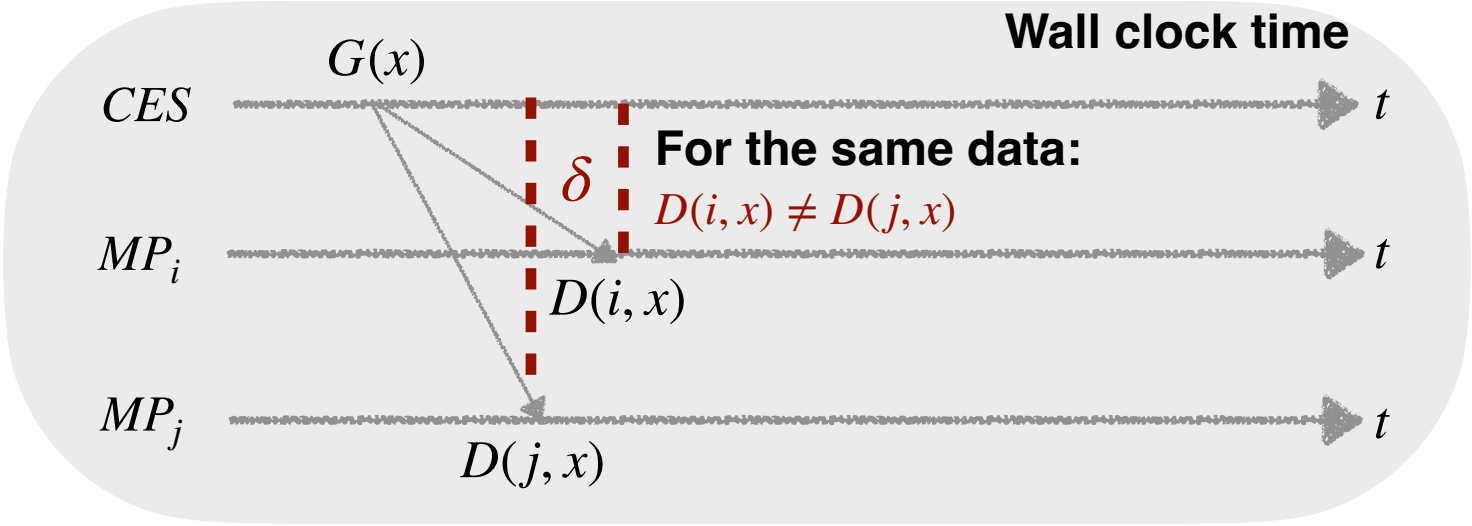
Abstraction

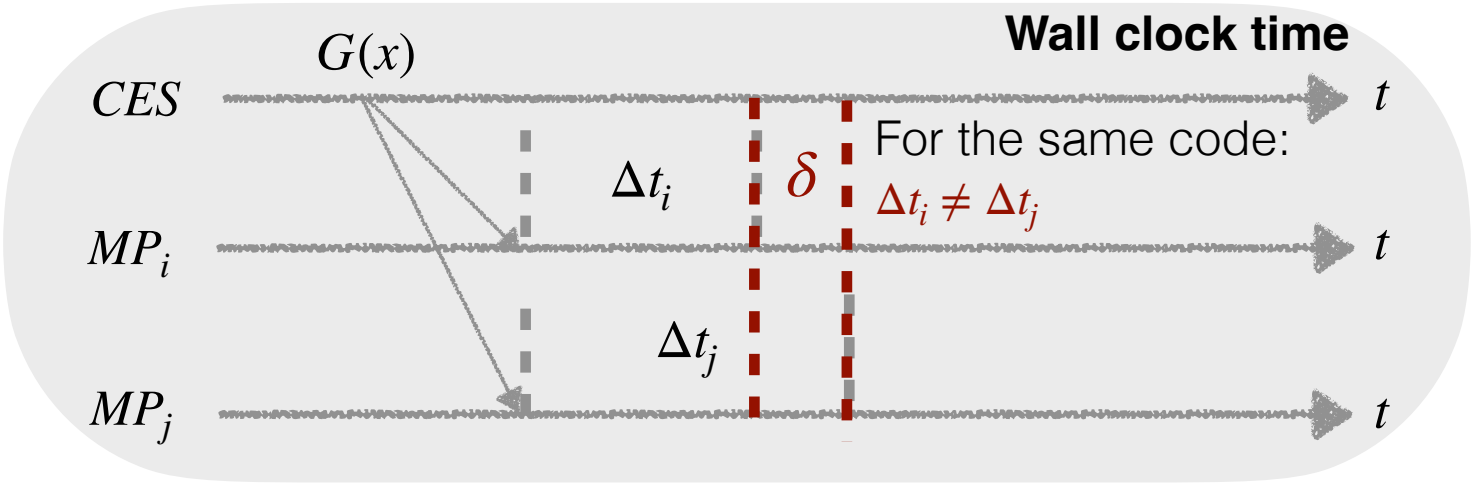
- Equal cloud networks
- Equal execution hardware
- …

Outline

- **Conceptual foundation**
- Implementing virtual time overlay
- Evaluation

Let's reflect on underlying model today…





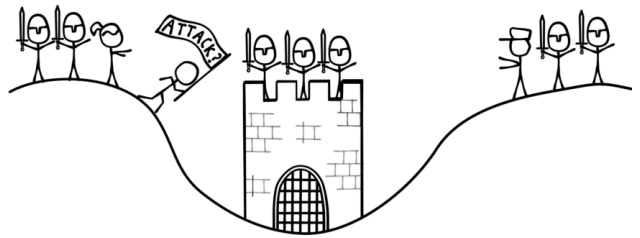


Image source: https://haydenjames.io/the-two-generals-problem/

Simultaneous delivery is ***infeasible***!

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

Let's reflect on underlying model today…

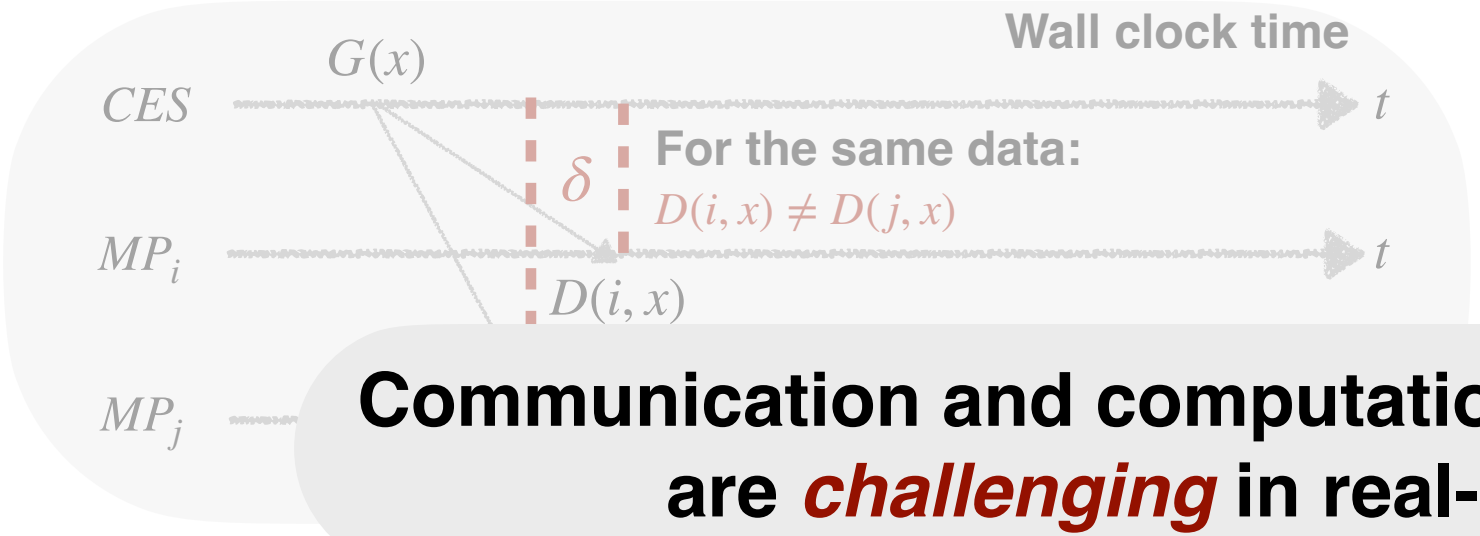
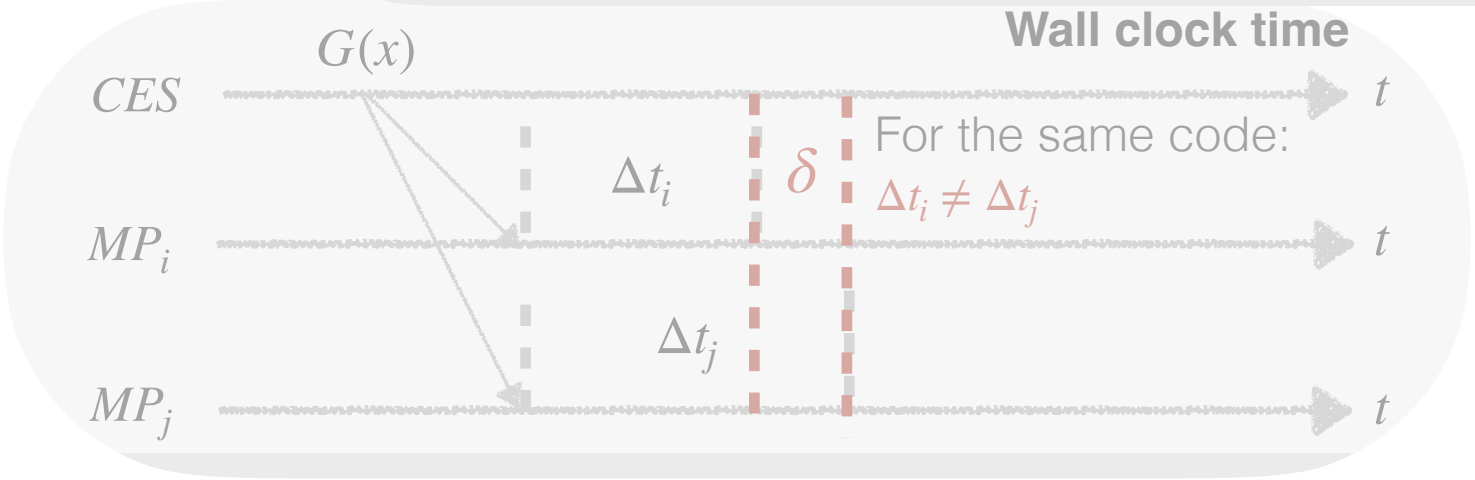


Image source: https://haydenjames.io/the-two-generals-problem/

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

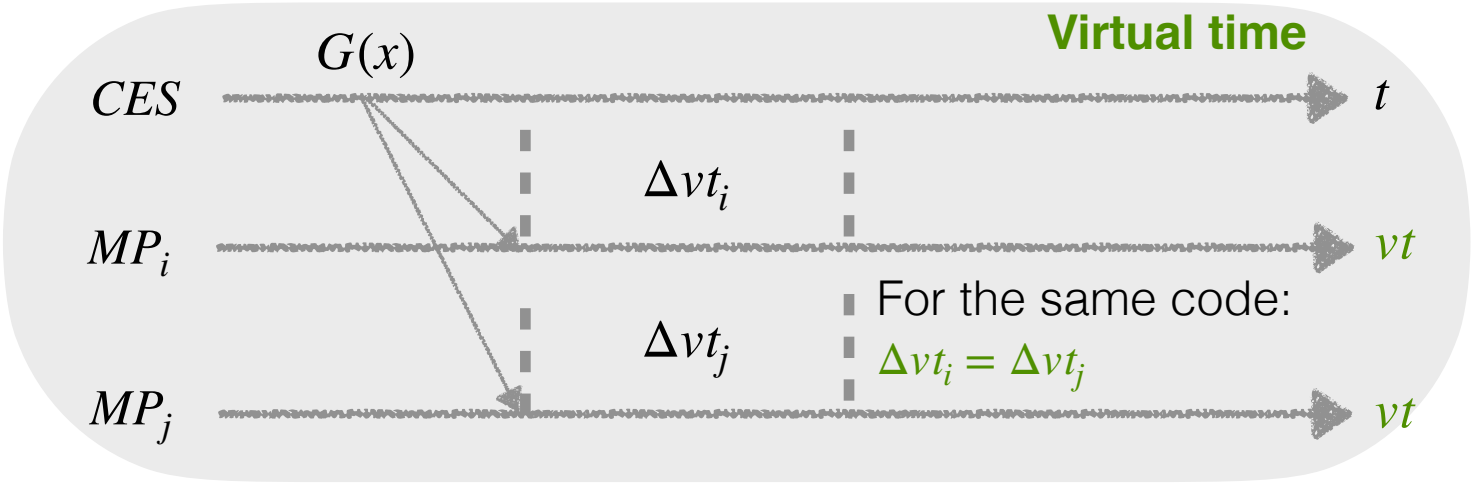
delivery
ible!



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

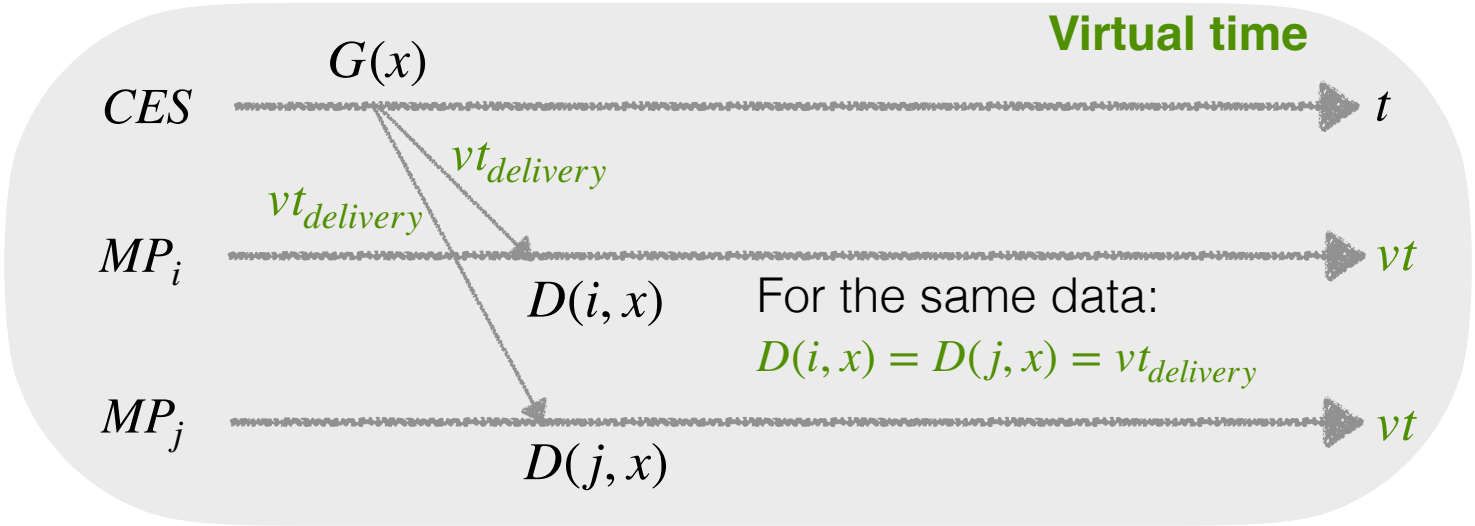
Let's try virtual time domain …

Virtual time unit ≡ *some equal amount of work*



Execution synchrony:

Advancing virtual time per ***'actual amount of work'***



Communication synchrony:

Releasing data to MPs at the ***same virtual delivery time***

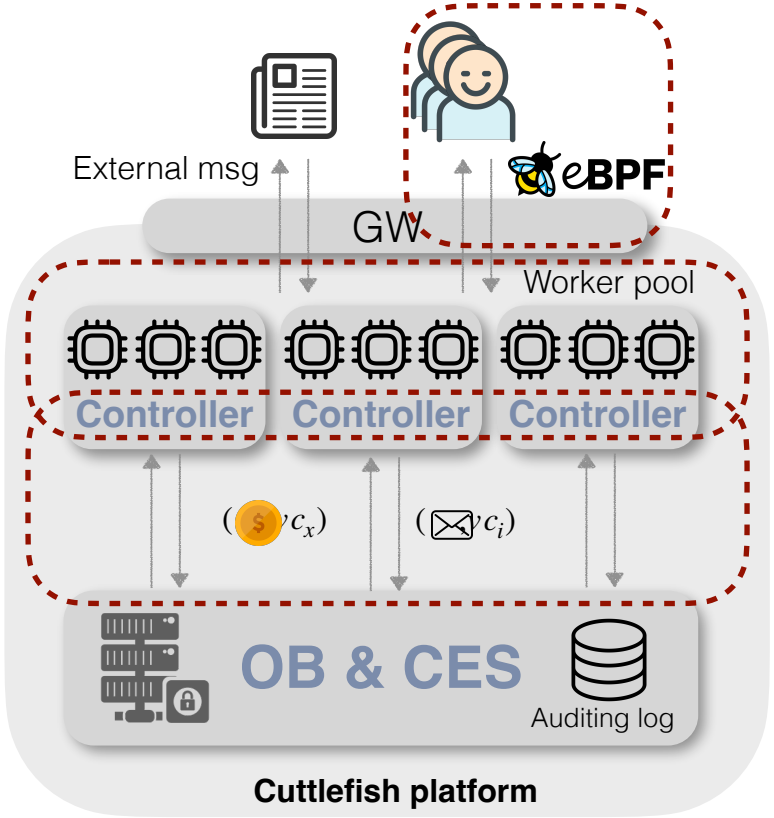
Outline

- Conceptual foundation
- **Implementing virtual time overlay**
- Evaluation

Implementing virtual time abstraction

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

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



1. **Programming interface**
3. **Runtime execution**
2. **Virtual cycle tracking**

User programming abstraction

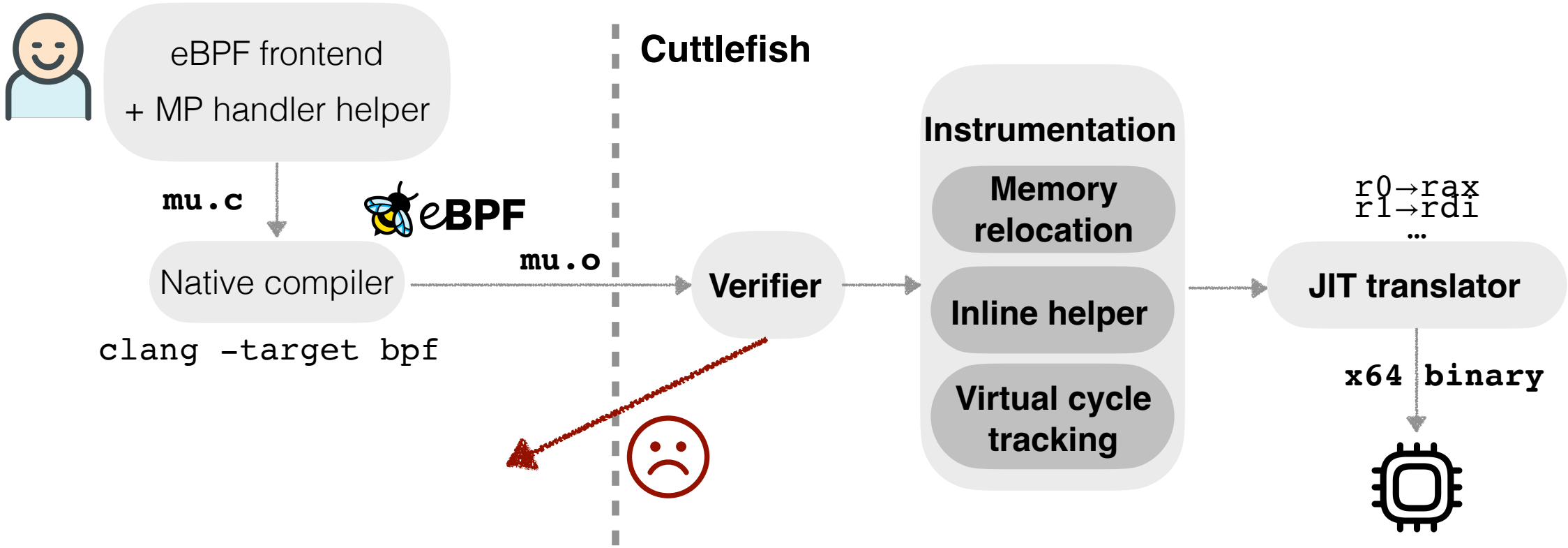
```
#include <cuttlefish_user.h>

int mp_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

MP code lifetime



2-tier compilation with the platform agnostic IR:
Track virtual cycle (fairly) in eBPF, but ***execute*** (efficiently) on native HW target

How to track and advance virtual time cycles?

eBPF asm

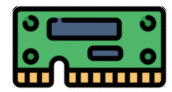
```
0000000000000000 <u_handler>:
       0:    85 00 00 00 0b 00 00 00    call 11
       1:    7b 0a f8 ff 00 00 00 00    *(u64 *)(r10 - 8) = r0
       2:    bf a2 00 00 00 00 00 00    r2 = r10
       3:    07 02 00 00 f8 ff ff ff    r2 += -8
       4:    18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 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

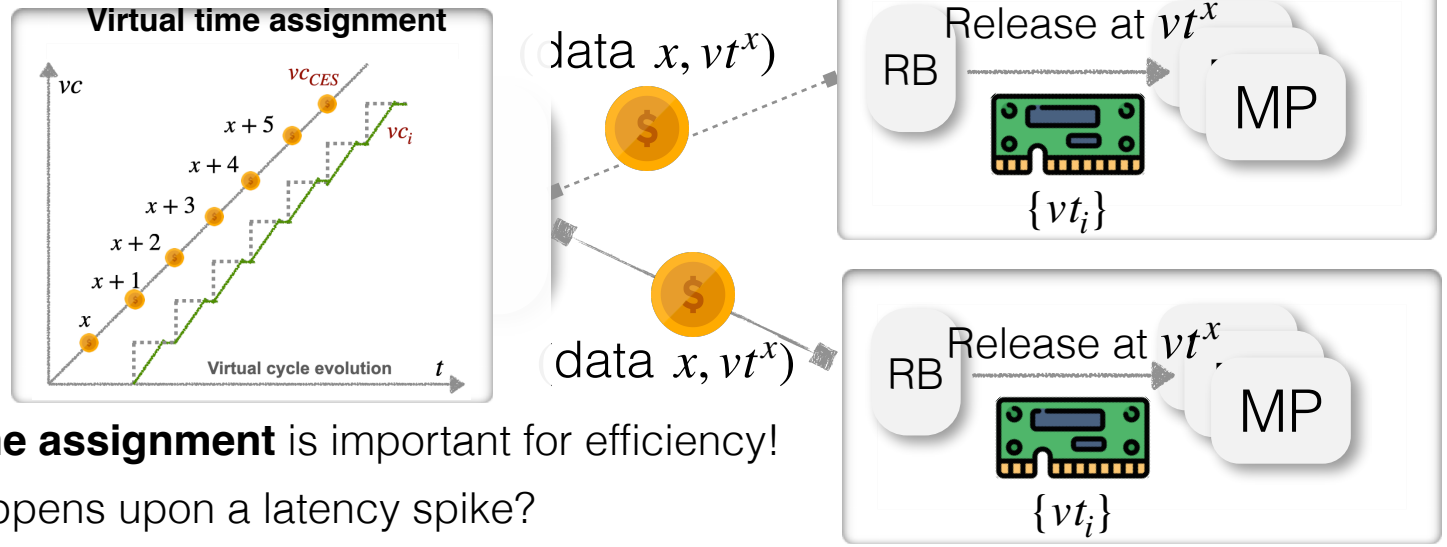
$vt_i += \Delta vt$



$\{vt_i\}$ maintained by execution runtime

- Break into basic blocks for batch updates of vt_i
  - JMP source, JMP destination, trade submission call
- Emit native machine code (2 x64 instr.) at the epilogue during JIT translation
  - Dummy trade/heartbeat for large blocks
  - Update the offsets for the (direct) JMP instructions

Simultaneous data delivery in virtual time



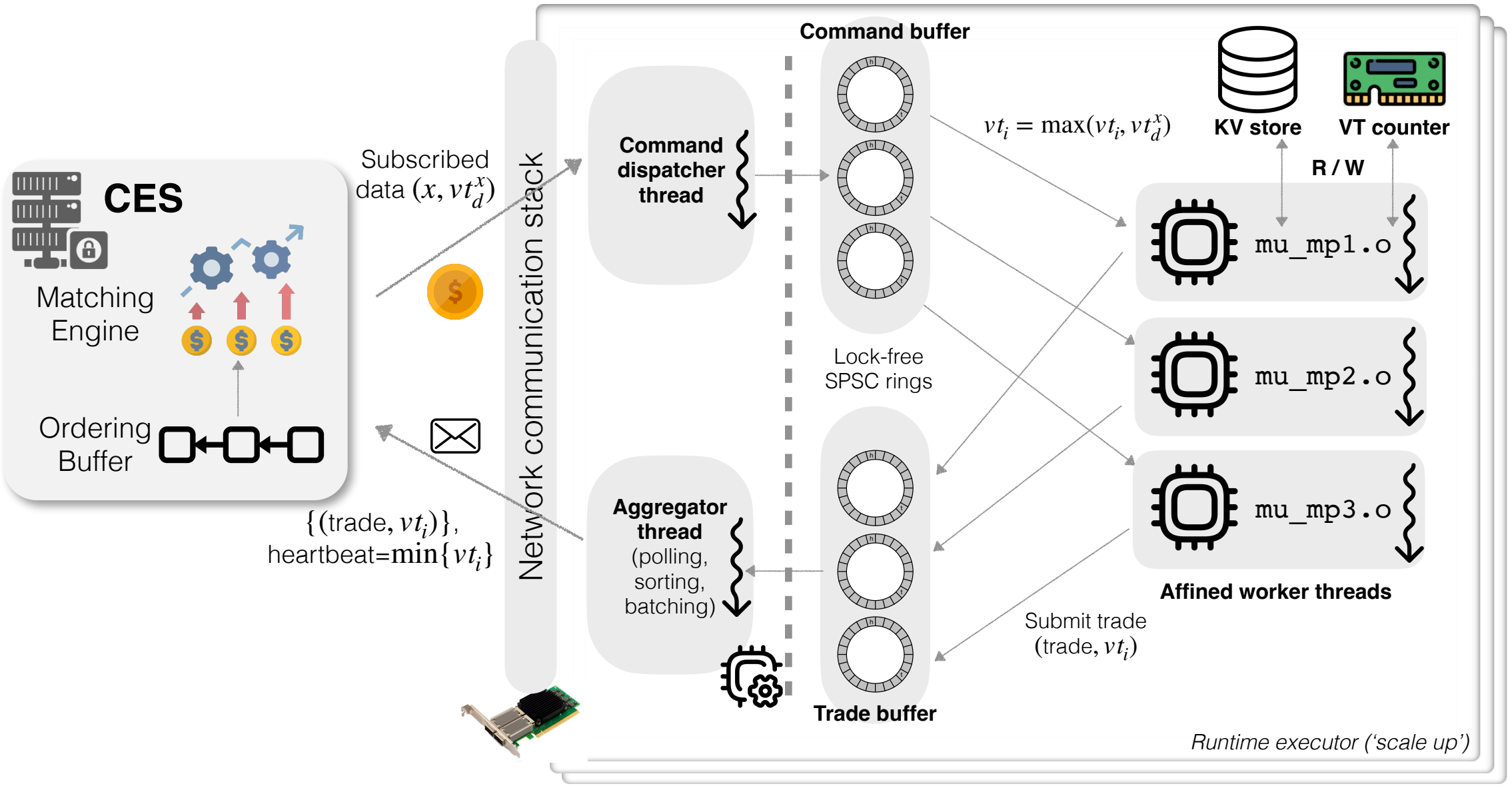
Virtual time assignment is important for efficiency!

- What happens upon a latency spike?
- What if some processor executions get slower?

More details:

- *Virtual time assignment algorithm*
- *Fault tolerance*
- *Handling external messages*
- *Security & trust discussions*

Runtime execution workflow



Outline

- Conceptual foundation
- Implementing virtual time overlay
- **Evaluation**

Comparison with existing ordering schemes

Ordering mechanisms

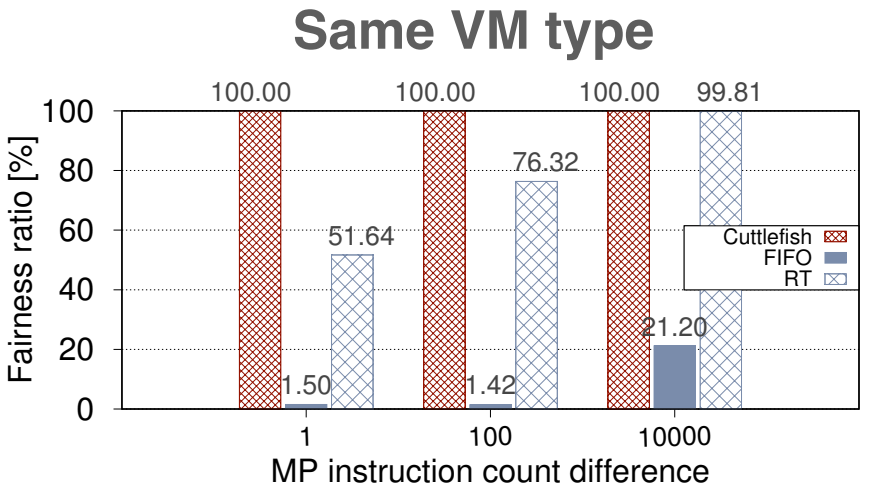
- Response Time (RT) based ordering
- FIFO ordering

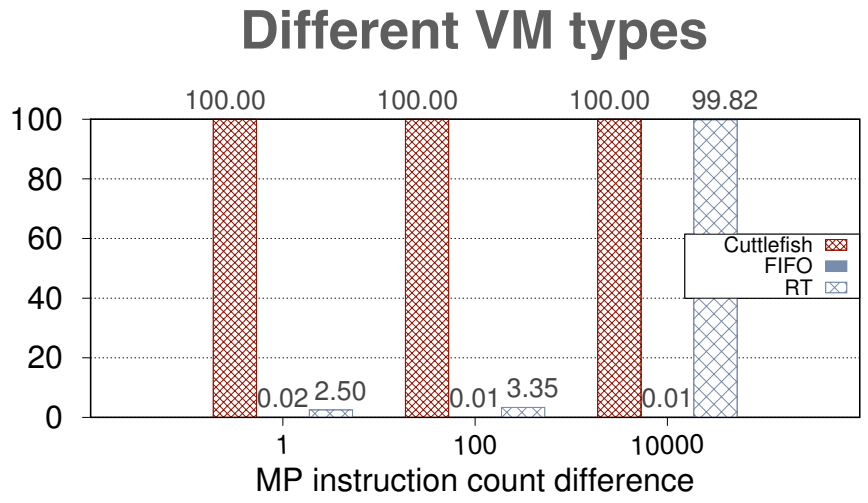
Set up

- Two MPs on two VMs
- MP_a executes N additional primitive IR instructions than MP_b
- Market data rate: every $\approx 100\mu s$

Metric

- Fairness ratio





Performance cost for fairness

Set up

- 100 MPs on 10 VMs
- Market data rate: every $\approx 100\mu s$
- CX-4 NIC and Intel Xeon Platinum 8272CL CPU @ 2.60GHz

	Latency (µs)				
	avg.	**p50**	**p90**	**p99**	**p99.9**
MaxRTT	52.04	47.74	49.95	55.85	144.2
Cuttlefish	54.19	50.82	53.49	68.46	166.3
	+2.15	**+3.08**	**+3.54**	**+12.61**	**+22.1**

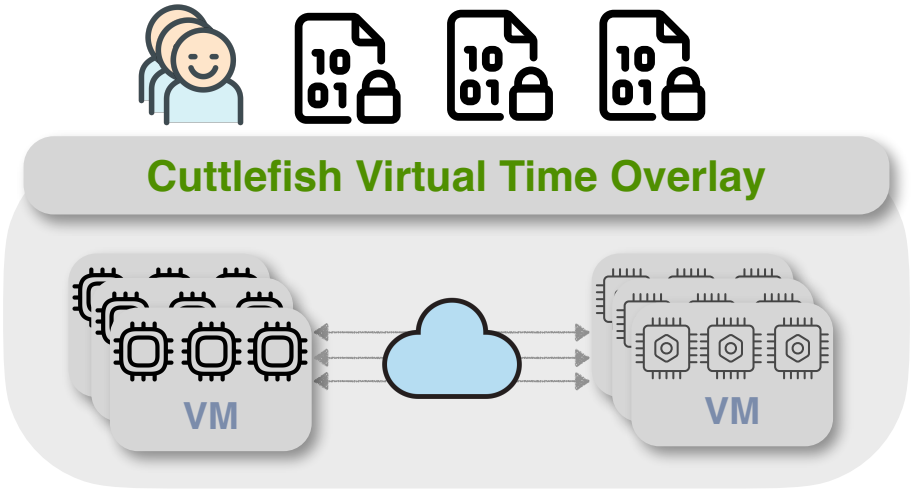
More details:

- *Execution throughput and latencies under processor disparities*
- *Virtual time instrumentation overhead*
- *Recovery under failures*

Summary

Cuttlefish: a fair, predictable cloud-hosted exchange platform

- Abstracting out variances in cloud communication and execution hardware
- An efficient implementation runnable on commercial cloud



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
- …