CloudEx: A Fair-Access Financial Exchange in the Cloud

Ahmad Ghalayini
Stanford University
Stanford, USA

Jinkun Geng
Stanford University
Stanford, USA

Vighnesh Sachidananda
Stanford University
Stanford, USA

Vinay Sriram
Tick Tock Networks
Palo Alto, USA

Yilong Geng
Tick Tock Networks
Palo Alto, USA

Balaji Prabhakar
Stanford University
Stanford, USA

Mendel Rosenblum
Stanford University
Stanford, USA

Anirudh Sivaraman
NYU
New York, USA

ABSTRACT

Financial exchanges have begun a move from on-premise and custom-engineered datacenters to the public cloud, accelerated by a rush of new investors, the rise of remote work, cost savings from the cloud, and the desire for more resilient infrastructure. While the promise of the cloud is enticing, the cloud's varying network latencies can lead to market unfairness: orders can be processed out of sequence, and market data can be disseminated to market participants at incorrect times due to varying latencies between participants and the exchange. We present CloudEx, a fair-access cloud exchange, which leverages high-precision software clock synchronization to compensate for noisy network conditions in the public cloud. We also discuss refinements to the CloudEx design that were informed by lessons learned from deploying CloudEx in two academic courses and conclude by outlining future research directions.

CCS CONCEPTS

• **Networks** → **Cloud computing**; • **Computer systems organization** → *Real-time systems*; • **Applied computing** → *Online auctions.*

KEYWORDS

financial exchanges, fair-access exchanges, high-frequency trading, low-latency systems, clock synchronization

ACM Reference Format:
Ahmad Ghalayini, Jinkun Geng, Vighnesh Sachidananda, Vinay Sriram, Yilong Geng, Balaji Prabhakar, Mendel Rosenblum, and Anirudh Sivaraman. 2021. CloudEx: A Fair-Access Financial Exchange in the Cloud. In *Workshop on Hot Topics in Operating Systems (HotOS '21), May 31–June 2, 2021, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3458336.3465278

1 INTRODUCTION

Equity markets provide financial services to hundreds of millions of market participants globally [5, 6]. In 2020 alone, these markets added 10 million new participants [13]. At the heart of these markets are electronic stock exchanges such as NASDAQ and NYSE.

These exchanges are required by regulation to provide market participants with *fair access* to the market [41]. Fair access has two aspects. First, when the exchange processes incoming orders from participants, it must execute them in the same sequence in which participants issued them. Second, when the exchange disseminates market data (e.g., stock quotes) to participants, it must do so simultaneously so that no participant has earlier access to market data [25, 32, 33, 39, 42]. Beyond regulation, market participants with lower latency can exploit short-lived mispricing in financial assets. Economists estimate that about 5 billion dollars are at stake annually from such latency arbitrage in global equity markets [20].

Today's exchanges use on-premise infrastructure that is heavily engineered to meet fair-access requirements. For instance, market participants are co-located in the same physical facility as the exchange. Further, they are connected to the exchange through cables whose lengths are equalized across participants [2]. This ensures that participant orders do not overtake each other during order processing and that market data arrives at participants nearly simultaneously.

Recently, exchanges have been contemplating moving their on-premise infrastructure to the cloud. This move has been motivated by the rapid increase in the number of participants, lower outages in the cloud, and cost savings from moving to the cloud [11]. The rise of remote work due to the COVID-19 pandemic has also added urgency to this cloud migration [11, 34]. For instance, NASDAQ plans to migrate all markets to the public cloud in the next decade [9, 11] and has already started offloading data processing to the cloud [12].

While migrating exchanges to the public cloud has several benefits, the best-effort nature of the cloud also poses challenges to meeting fair-access requirements. For instance, in contrast to the deterministic exchange-participant latencies provided by carefully engineering network paths in on-premise exchanges, exchange-participant latencies in the cloud can vary unpredictably over time. This variation in network latencies causes unfairness: it allows some participants' orders to overtake others en route to the exchange and allows market data to reach some participants earlier than others.

To understand the technical challenges of building a fair-access financial exchange in the cloud, in collaboration with a major stock exchange, we are building CloudEx—to our knowledge, the first demonstration of a fair-access exchange in the cloud. We have so far evaluated CloudEx in realistic settings using student participants in two courses. The learnings from both courses have helped us further refine CloudEx for real-world use.

CloudEx currently incorporates three main ideas. First, it uses high precision (nanosecond-level) software-based clock synchronization [30] to measure variable VM-to-VM latencies in the cloud. Variable message latencies can be compensated by delaying messages arriving earlier by a certain time d to ensure fairness (§2).

Second, in our course deployment of CloudEx, we observed that network latencies in the cloud change over time and with the number of participants. Thus, a fixed d will result either in higher-than-needed latencies or unfairness at some point. Hence, we developed a control strategy to tune d to achieve a target unfairness ratio, regardless of changing cloud conditions (§3).

Third, in our course deployment, we also observed high order submission latencies because of slow or failed gateways that connect participants to the exchange. For fault tolerance and to reduce latency, especially at the tail, orders in CloudEx are replicated and routed to the exchange through multiple gateway VMs. This allows the exchange to pick the earliest arriving order (§3).

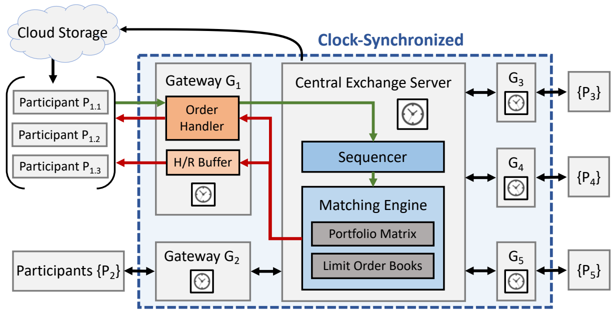


Figure 1: System Architecture of CloudEx

We conduct a preliminary evaluation of CloudEx in §4. Our key takeaway from building CloudEx is that it is possible to build a fair-access stock exchange in the cloud. However, CloudEx has not yet been optimized for performance and its current throughput and latency (Table 1) fall short of on-premise exchanges (60−100K orders/second on average[1] and ~60 µs [10, Page 21]). We discuss ongoing optimizations to improve CloudEx's performance (§6) and outline other areas for future work (§7).

2 DESIGN

2.1 System Architecture

CloudEx consists of market participants, gateways, a central exchange server, and cloud storage. Fig. 1 summarizes the CloudEx architecture. Fig. 2 illustrates the lifecycle of an order in CloudEx. We use ZeroMQ [17] for reliable network communication between the different components. Each gateway clock is precisely synchronized to the reference clock at the central exchange server.

Market Participants. Each market participant owns a VM that is connected to one of the gateways. The participants are provided APIs that allow them to (1) submit orders and receive order and trade confirmations, (2) subscribe to real-time market data streams, and (3) query for historical market data from a long-term cloud storage module. To place an order, a participant constructs an order message and forwards it to the participant's assigned gateway. An order message contains the symbol to be traded (e.g., a stock or future contract), the action (buy or sell), the number of shares, the order type (e.g., limit order [14] or market order [15]), and the limit price for limit orders.

Gateways. The gateways sit between the market participants and the central exchange server, routing orders to the central exchange server and routing market data to the participants [20]. Gateways are also required to secure the matching engine from abuse, e.g., unauthenticated or invalid orders. Each gateway hosts two modules: an order handler and a hold/release (H/R) buffer. The order handler authenticates

[1]Based on our conversations with exchange operators.

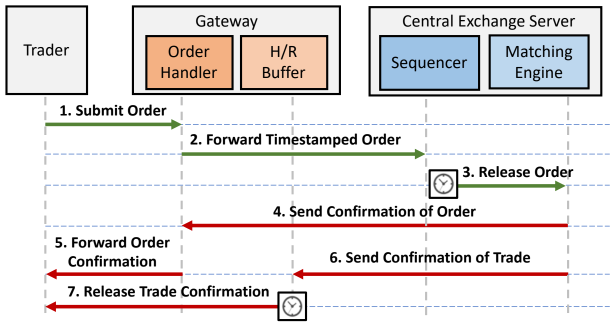

Trader
Gateway
Order Handler
H/R Buffer
Central Exchange Server
Sequencer
Matching Engine
1. Submit Order
2. Forward Timestamped Order
3. Release Order
4. Send Confirmation of Order
5. Forward Order Confirmation
6. Send Confirmation of Trade
7. Release Trade Confirmation

Figure 2: Lifecycle of an Order in CloudEx

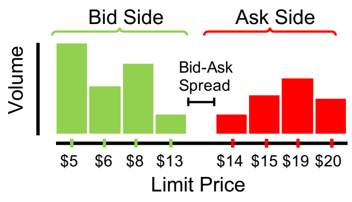


Figure 3: Visualization of a Limit Order Book

and validates orders received from the participants, and then assigns a globally synchronized timestamp to these orders before forwarding them to the central exchange server.[2] The order handler is also responsible for routing order confirmations received from the central exchange server back to the participants (Fig. 2). Meanwhile, real-time market data received from the central exchange server is dispensed to each participant via the H/R buffers. Each H/R buffer *holds* a given piece of data until the prescribed *release* time for that piece of data. Assuming market data arrives early enough, precise clock synchronization and identical release times guarantee that market data is simultaneously sent to all participants.

Sequencer. The sequencer receives inbound orders from the gateways and enqueues each order into a priority queue based on the order's gateway-assigned timestamp.

Matching Engine. The matching engine dequeues orders from the sequencer and runs the "continuous price-time matching" algorithm [8], used by most exchanges. The algorithm maintains two data structures: limit order books (Fig. 3) that store unmatched limit orders, and a portfolio matrix that tracks each market participant's assets and cash balance. The limit order book for each symbol stores the bids (buy orders) and asks (sell orders) for that symbol. The bids are sorted in order of decreasing limit price, and the asks sorted in order of increasing limit price. A match, which produces one or more trades, happens when either an incoming bid (ask) specifies a price greater (respectively, less) than or equal to the lowest ask (respectively, highest bid) in the order book. Incoming limit orders that cannot be matched on arrival are added to the book. If multiple incoming limit orders at the same price are matched against a bid/ask in the limit order book, ties are broken based on the orders' gateway timestamps, with priority given to the earlier timestamps.

After each order is processed, an order confirmation is generated and sent to the appropriate gateway's order handler (Fig. 2). When a trade is executed, the portfolio matrix is updated, and a record of the trade is generated. Trade records consist of the traded symbol, the number of shares traded, and the execution price, and are persisted in Google Bigtable [26]. Market participants are provided an API to query historical market data from Bigtable.

Market Data Dissemination. The final function of the matching engine is to provide real-time market data streams to the participants via the H/R buffers at the gateways. Market data includes both trade records and periodic snapshots of the limit order books. Market participants subscribe to this data per symbol. Before trade records and limit order book snapshots are forwarded to the gateways, the matching engine assigns a release timestamp that specifies when the gateways ought to dispense each piece of market data to the participants (Fig. 2).

2.2 Unfairness: Definitions and Remedies

In practice, it is impossible to build a perfectly fair exchange because of the variable delays from the gateways to the matching engine. We quantify how unfair an exchange is through (1) an inbound unfairness ratio and (2) an outbound unfairness ratio. In an ideal exchange, both should be zero. To remedy each source of unfairness, we introduce a compensating delay parameter. We explain the unfairness ratios and delay parameters below.

Inbound Unfairness Ratio. Inbound unfairness ratio is defined as the percentage of total orders that are processed out of sequence. An order is considered to be processed out of sequence when its gateway-assigned timestamp is earlier than that of the preceding *processed* order.

Outbound Unfairness Ratio. Outbound unfairness ratio is defined as the percentage of total pieces of market data that are unfairly disseminated. A piece of market data is considered to be unfairly disseminated if one or more gateways receive it later than the data's designated release time.

Sequencer Delay Parameter, d_s. To address inbound (order) unfairness, the sequencing buffer is instructed to hold orders until a delay d_s has elapsed relative to their gateway timestamps. In other words, the sequencer dequeues an order O if and only if $t_C - t_O \geq d_s$, where t_C represents the current time and t_O represents the gateway timestamp of the order. This delay provides orders submitted before O

[2]Order timestamps are taken at gateway VMs instead of the participant VMs for security: gateway VMs are owned and managed by the exchange operator, unlike participant VMs. The timestamps could also be taken in a secure enclave [7] within participant VMs.

sufficient time to arrive at the central exchange server and be properly sequenced by the priority queue before order O itself is dequeued.

Hold/Release Buffer Delay Parameter, d_h. To address outbound (market data) unfairness, we use hold/release buffers. Consider a piece of market data M, and let t_M represent its creation time at the matching engine. The hold/release buffer at each gateway is instructed to release M at time t_R, where $t_R = t_M + d_h$. d_h is the maximum time that a piece of market data has to wait in the hold/release buffer before it can be released to the market participants. This delay provides all gateways enough time to receive M from the matching engine before M's release time.

Consequences of d_s and d_h. The values of d_s and d_h control the latency-fairness trade-off (Fig. 4) in CloudEx. Large delay parameter values cause the system to become less responsive to market participants due to larger latencies, while small values make the system less fair.

3 COURSE DEPLOYMENTS AND REFINEMENTS

To understand the challenges of operating CloudEx, we deployed it in two courses using students as participants. We discuss learnings from these deployments and how we refined CloudEx based on them.

Deployment Setup. For our first deployment, we held a 3-hour trading competition between 13 trading groups of high school students as part of a summer course. Over this competition, students placed roughly 1000 orders and 1000 trades across 10 symbols. For each symbol we initiated trading bots to place trades to induce specific price-time patterns on which students could engineer algorithms. For our second, larger, deployment, we deployed CloudEx in a financial technologies and algorithms course attended by undergraduate, masters, and PhD students. Over 8 trading days, 32 groups of students placed 4.2 million orders and 330000 trades.

Learnings from Deployments. Our first deployment led to changes in our APIs to make CloudEx more user-friendly. It also led us to the observation that we could support more symbols in CloudEx by sharding the matching engine over symbols. In our second deployment, we observed that with a fixed delay parameter, the unfairness ratios varied daily due to changing cloud network conditions. This led us to develop a control scheme (DDP) to tune delay parameters automatically based on sensed network conditions. We also observed that straggler and faulty gateways can inflate latency, leading us to a scheme (ROS) to replicate orders through multiple gateways for lower latency and fault tolerance.

Sharding. We shard the matching engine based on symbols, with each shard dequeuing orders from its own order priority queue and managing the limit order books of a subset of symbols. Based on its symbol, an order is routed to the corresponding shard.

Dynamic Delay Parameters (DDP). Empirically, in our deployments, we observed that there is no simple relationship between the unfairness ratios and the delay parameters (Figs. 4 and 5 further illustrate this). Hence, we cannot easily derive and directly control the unfairness ratio by setting the delay parameter to a static value. Further, time-varying latencies in the cloud necessitate that these parameters should be updated *continuously* to achieve a target unfairness ratio. To directly control the unfairness ratio, CloudEx continuously and independently tunes each delay parameter. For each parameter, DDP uses a rolling window of order/market data samples (of size 1000 samples/window) to calculate the unfairness ratios in real time. If the current unfairness ratio is above the target unfairness ratio, DDP increases the delay parameter by a small fixed amount (5 µs), else DDP decreases it by the same amount.

Replicated Order Submission (ROS). VMs are not homogeneous and stragglers are common in the cloud [27]. When market participants submit orders to a slow gateway, they suffer from a long delay before receiving an order confirmation. Even worse, if a gateway crashes, the corresponding participants will not be able to submit any orders until that gateway restarts. To address such stragglers and achieve fault tolerance, we use message replication by letting market participants submit replicas of the same order through *multiple* gateways instead of one gateway. The matching engine processes the earliest-arriving replica and drops the others.

4 EVALUATION

Testbed. To benchmark CloudEx and evaluate the effectiveness of DDP and ROS, we launch a 65-node Google Cloud cluster, including 48 market participant VMs of type `n1-highmem-2`, 16 gateway VMs of type `n1-highmem-8` and 1 matching engine VM of type `n1-standard-64` [18]. All VMs are deployed within the same zone (`us-central1-a`).

Trading Setup. The exchange has 100 symbols for trading. For §4.2, the matching engine has one shard, and each market participant submits around 450 orders/s on average (22K orders in total processed by the matching engine per second), for a total duration of 5 minutes per experiment.

Clock Synchronization. The VM clocks are synchronized using the Huygens algorithm [30]. During a sample 3-hour run in our testbed, the 99^{th} percentile clock offsets average around 159 ns.[3] To illustrate the benefits of synchronization, we first run CloudEx without the inbound resequencing mechanism. The inbound unfairness ratio is 24.6%. With clock synchronization, even a static d_s of 0 achieves an inbound unfairness ratio of 8.4%.

4.1 Throughput and Latency

We benchmarked our matching engine by varying the number of shards; throughput and median latency results are summarized in Table 1. After reaching 8 shards, the throughput stops improving since different shards need to serialize their updates to shared data structures (e.g., portfolios).

Table 1: CloudEx Throughput and Median Latency as a Function of Number of Shards

Shards	Throughput	Submission Latency (μs) [1]	End-to-End Latency (μs) [2]
1	22k	365	1128
2	40k	402	1089
4	49k	401	1094
8	61k	390	1080
16	61k	395	1044

[1] The submission latency is the latency from when the participant submits the order to when the matching engine receives the order.

[2] The end-to-end latency is the latency from when the participant submits the order to when it receives the order confirmation from the matching engine.

4.2 Evaluation of DDP and ROS

DDP Evaluation. Figs. 4 and 5 demonstrate the advantages of the DDP strategy over static delay parameters of different values from 400 − 1200 μs. DDP enjoys two major benefits. (1) *Direct control over unfairness ratios*: Instead of setting a delay threshold that has a complex relationship with unfairness ratio, we can set a target unfairness ratio. Then, DDP can automatically tune delay parameters to meet this target. This makes the unfairness ratio more predictable. (2) *Adaptability*: When the message delay varies during runtime, DDP dynamically tunes the delay value to guarantee a fairness target, without sacrificing too much queuing/releasing delay.[4]

We evaluate two scenarios to prove each benefit. In the first scenario, we simply run CloudEx without artificial delays, to prove DDP can control the unfairness ratio according to different targets (Fig. 4). For the second one, to examine the adaptability of DDP, we simulate variable latencies by periodically injecting 0, 400 and 200 μs of delays to the gateway-engine link every 6 seconds (Fig. 5).

Fig. 4 indicates that the DDP strategy (shown in red) offers direct control, allowing exchange operators control over the level of fairness for market participants. As shown in the figure, all experiments with DDP yield unfairness ratios that are very close to the target ratios. By contrast, sweeping static delay parameters (shown in blue) achieves a poorer trade-off, because unfairness ratios and delay parameters are not simply related. For example, in some cases, a small reduction of the delay parameter (e.g., from 400 μs to 200 μs for the inbound direction) may lead to a worse unfairness ratio by more than one order of magnitude. In other cases, a large increase (e.g., from 400 μs to 1000 μs for the inbound direction) offers little reduction in the unfairness ratio, illustrating the difficulty of directly setting delay parameters.

Fig. 5 indicates that DDP adapts to the time-varying environment, and achieves a better trade-off between fairness and queuing/releasing delay. In other words, with similar inbound/outbound unfairness ratio, DDP achieves lower queuing/releasing delay than the static counterparts.

ROS Evaluation. In Fig. 6a, we show the change of the submission latency as the replication factor (RF) increases. Comparing the single submission (RF = 1) with the replicated submission (RF = 3), we can see that the median latency is reduced by 15% (365 μs to 309 μs), and the 99.9th percentile latency is reduced by 40% (1096 μs to 658 μs). However, increasing the RF also increases CPU usage (Fig. 6b). When the RF exceeds 3, latency degrades due to the CPU spending more time in discarding duplicates.

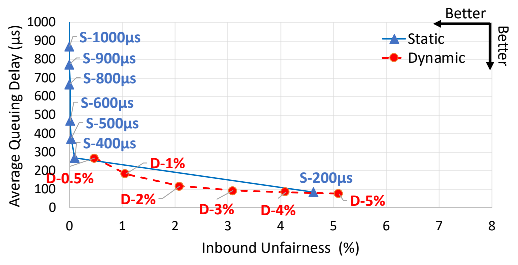
5 RELATED WORK

Algorithmically Tackling Unfairness. Complementary to CloudEx's goal of reducing latency variability, several proposals tackle unfairness due to latency variability by changing the matching engine's algorithm itself. For instance, some order-matching algorithms give equal [25] or random [36] matching priority to orders arriving within a certain *time range*—instead of the strict FCFS order required by continuous price-time matching [8].

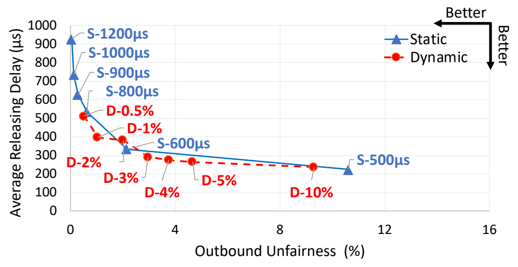
Online Multiplayer Games. Multiplayer online gaming servers, like CloudEx's exchange servers, ideally process players' actions in the order in which players executed them, and publish game state updates to all the players simultaneously. Sync-MS [35] similarly uses synchronized timestamps for fairness, but is evaluated in simulation instead of in the cloud. Moreover, timescales in the gaming context (milliseconds) are an order of magnitude larger than those in financial exchanges (∼100 μs).

[3]When we tried CloudEx with NTP [37], the standard in software clock synchronization, we found ~10 ms clock offsets between gateways. These offsets are much larger than CloudEx's gateway-to-matching-engine latencies, making NTP unsuitable for CloudEx.

[4]Queuing delay is the time between the enqueue and the dequeue of an order at the sequencer. Releasing delay is the time between the hold and the release of one piece of market data at the H/R buffer.



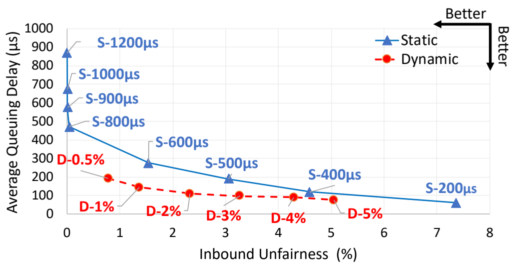
(a) Inbound



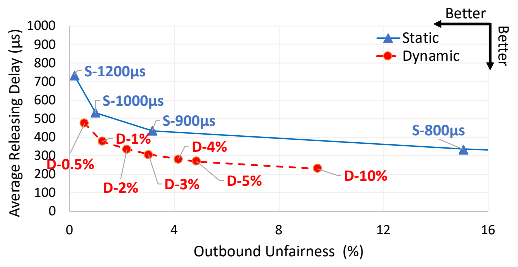
(b) Outbound

Figure 4: Evaluation of DDP (No Artificial Delay) - 48 Market Participants, 22k orders/sec, 5 Minute Duration

The blue point S-xµs indicates a static delay parameter of x µs. The red point D-y% indicates a DDP strategy with a target unfairness ratio of y%. The same notation also applies to Fig. 5.

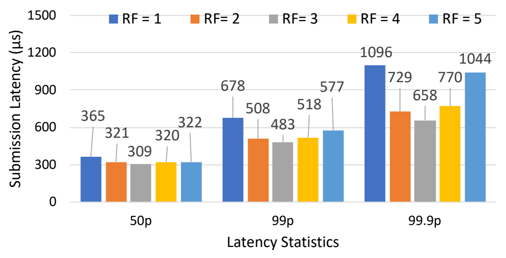


(a) Inbound

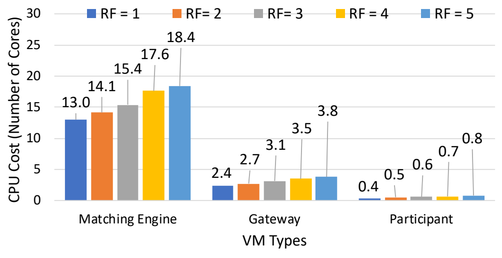


(b) Outbound

Figure 5: Evaluation of DDP (With Artificial Delay) - 48 Market Participants, 22k orders/sec, 5 Minute Duration, Periodic Artificial Delays of 0, 400, or 200 µs Every 6 Seconds



(a) Submission Latency



(b) CPU Cost

Figure 6: Evaluation of ROS - 48 Market Participants, 22k orders/sec, 5 Minute Duration

6 DISCUSSION

We now discuss security, regulatory, and performance concerns associated with CloudEx and also outline avenues for future work on CloudEx.

Security. We envision that addressing security concerns in migrating an exchange to the public cloud would be best facilitated by a tight feedback loop between exchange operators and cloud providers. AWS GovCloud [22] and Azure for US Government [23] offer insights into practices deployed when migrating infrastructure that hosts highly sensitive and access-controlled information. For example, GovCloud capabilities include (1) limiting personnel with access to physical infrastructure, (2) auditing the logs of accesses to VMs, and (3) controlling a cloud customer's access to specific resources using identity management [21].

Regulation. Exchanges vary in the extent of their regulation. At the one extreme, exchanges for cryptocurrencies (e.g., Binance [24] and Coinbase [28]) and auction/bidding platforms (e.g., eBay) are relatively unregulated, while exchanges for equities (e.g., NASDAQ) are subject to regulations by government agencies (e.g., FINRA and the SEC in the US). These regulations are location-dependent and include

Reg NMS in the US [40] and MiFID II [38] in Europe. While CloudEx's ideas are applicable to all of these exchanges, the more unregulated exchanges do present an easier path to the short-term adoption of CloudEx. For exchanges where regulation is critical, we note that several other functions in the finance industry and other closely regulated industries such as healthcare have successfully migrated to the cloud and these can potentially serve as a blueprint for financial exchanges as well [29].

Performance. We identify a few areas of performance improvement in CloudEx. Networking primitives like DPDK [3] and RDMA [31] can be used instead of ZeroMQ to lower latency; throughput can be improved by employing fine-grained locking of shared data structures; and unfairness caused by variability in latency measurements can be reduced by using NIC hardware timestamps [1, 4]. The use of bare-metal instances in the cloud can further improve CloudEx's performance.

7 CONCLUSION

Exchanges have started migrating from carefully-designed and bespoke infrastructure to the noisy and best-effort public cloud. In this paper, we present CloudEx—a fair-access cloud exchange. We are currently pursuing three main avenues for future research. First is improving CloudEx's performance as described in §6. Second, we plan to use CloudEx as a market simulator for conducting research on exchange design (e.g. new auction mechanisms [25] and order types [16]). Third, we believe the techniques in CloudEx are applicable beyond financial exchanges to domains such as ad exchanges [19], massively multiplayer games, and multi-agent control. We also intend to open source CloudEx as we envision its most compelling use cases will be through its use as an open platform for teaching and research.

REFERENCES

[1] Azure: Accelerated Networking. https://azure.microsoft.com/en-us/blog/maximize-your-vm-s-performance-with-accelerated-networking-now-generally-available-for-both-windows-and-linux/. Accessed: 2021-02-02.

[2] Beyond Flash Boys: Improving Transparency and Fairness in Financial Markets. http://video.cfainstitute.org/services/player/bcpid3577743869001?bckey=AQ~~,AAABE5oc3_E~,Leu10fA0D1sc9Dh9wz3oyrstQJ-PkzpJ&bctid=4436841897001. Accessed: 2021-02-02.

[3] DPDK-Data Plane Development Kit. https://www.dpdk.org/. Accessed: 2021-02-02.

[4] Enhanced Networking on Linux - Amazon Elastic Compute Cloud. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html. Accessed: 2021-02-02.

[5] Fast Facts about Vanguard. https://about.vanguard.com/who-we-are/fast-facts/. Accessed: 2021-02-02.

[6] Fidelity Investments: Q2 Highlights. https://www.fidelity.com/bin-public/060_www_fidelity_com/documents/about-fidelity/corporate-statistics-infographic.pdf. Accessed: 2021-02-02.

[7] Intel® Software Guard Extensions. https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html. Accessed: 2021-02-02.

[8] Matching Orders Definition. https://www.investopedia.com/terms/m/matchingorders.asp. Accessed: 2021-02-02.

[9] Nasdaq Believes the Cloud Is the Future of Markets | Nasdaq. https://www.nasdaq.com/articles/nasdaq-believes-the-cloud-is-the-future-of-markets-2020-06-24. Accessed: 2021-02-02.

[10] Nasdaq Colocation. http://web.stanford.edu/class/cs349f/slides/CS349F_lec7_deutsche_precise_timing_in_finance.pdf. Accessed: 2021-02-02.

[11] Nasdaq Ramps Up Cloud Move-All 28 of the Company's Markets Are Expected to Be Hosted in the Cloud within the Next Decade. https://www.wsj.com/articles/nasdaq-ramps-up-cloud-move-11600206624. Accessed: 2021-02-02.

[12] Nasdaq Tech Chief Credits Cloud With Helping Manage Market Frenzies. https://www.wsj.com/articles/nasdaq-tech-chief-credits-cloud-with-helping-manage-market-frenzies-11611962716. Accessed: 2021-02-02.

[13] New Army of Individual Investors Flexes Its Muscle. https://www.wsj.com/articles/new-army-of-individual-investors-flexes-its-muscle-11609329600. Accessed: 2021-02-02.

[14] SEC.gov - Limit Orders. https://www.sec.gov/fast-answers/answerslimithtm.html. Accessed: 2021-02-02.

[15] SEC.gov - Market Orders. https://www.sec.gov/fast-answers/answersmktordhtm.html#:~:text=A%20market%20order%20is%20an,be%20executed%20is%20not%20guaranteed. Accessed: 2021-02-02.

[16] Self-Regulatory Organizations; Investors Exchange LLC; Order Approving a Proposed Rule Change to Add a New Discretionary Limit Order Type Called D-Limit. https://www.sec.gov/rules/sro/iex/2020/34-89686.pdf. Accessed: 2021-02-02.

[17] ZeroMQ. https://zeromq.org/. Accessed: 2021-02-02.

[18] Machine Types | Compute Engine Documentation | Google Cloud. https://cloud.google.com/compute/docs/machine-types, 2020. Accessed: 2021-02-02.

[19] Sebastian Angel and Michael Walfish. Verifiable Auctions for Online Ad Exchanges. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 195–206, 2013.

[20] Matteo Aquilina, Eric B Budish, and Peter O'Neill. Quantifying the High-Frequency Trading "Arms Race": A Simple New Methodology and Estimates. *Chicago Booth Research Paper*, (20-16), 2020.

[21] AWS. Amazon GuardDuty. https://aws.amazon.com/guardduty/. Accessed: 2021-05-09.

[22] AWS. AWS GovCloud. https://aws.amazon.com/govcloud-us/?whats-new-ess.sort-by=item.additionalFields.postDateTime&whats-new-ess.sort-order=desc. Accessed: 2021-05-09.

[23] Azure. Azure for US Government. https://azure.microsoft.com/en-us/global-infrastructure/government/. Accessed: 2021-05-09.

[24] Binance. Bitcoin Exchange | Cryptocurrency Exchange | Binance. https://www.binance.com. Accessed: 2021-05-09.

[25] Eric Budish, Peter Cramton, and John Shim. The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.

[26] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2), June 2008.

[27] James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Gregory R. Ganger, Garth Gibson, Kimberly Keeton, and Eric Xing. Solving the Straggler Problem with Bounded Staleness. In *14th Workshop on Hot*

Topics in Operating Systems (HotOS XIV), Santa Ana Pueblo, NM, May 2013. USENIX Association.

[28] Coinbase. Coinbase. https://www.coinbase.com. Accessed: 2021-05-09.

[29] Forbes. Many Highly Regulated Companies Have Made the Leap to the Cloud — You Can, Too. https://www.forbes.com/sites/forbestechcouncil/2021/04/30/many-highly-regulated-companies-have-made-the-leap-to-the-cloud-you-can-too/?sh=43f25d7a192d. Accessed: 2021-05-09.

[30] Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosunblum, and Amin Vahdat. Exploiting a Natural Network Effect for Scalable, Fine-Grained Clock Synchronization. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, pages 81–94, Berkeley, CA, USA, 2018. USENIX Association.

[31] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. RDMA over Commodity Ethernet at Scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 202–215, New York, NY, USA, 2016. Association for Computing Machinery.

[32] Neil Johnson, Guannan Zhao, Eric Hunsader, Jing Meng, Amith Ravindar, Spencer Carran, and Brian Tivnan. Financial Black Swans Driven by Ultrafast Machine Ecology. *arXiv preprint arXiv:1202.1448*, 2012.

[33] Brandon Keim. Nanosecond Trading Could Make Markets Go Haywire. *Wired, February*, 16, 2012.

[34] Kaplan Ken. Keeping Financial Traders Connected Remotely During the COVID-19 Crisis. https://www.nutanix.com/theforecastbynutanix/industry/how-jm-finn-kept-remote-stock-traders-connected-during-covid-19. Accessed: 2021-02-02.

[35] Yow-Jian Lin, Katherine Guo, and Sanjoy Paul. Sync-MS: Synchronized Messaging Service for Real-Time Multi-Player Distributed Games. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pages 155–164. IEEE, 2002.

[36] Vasilios Mavroudis and Hayden Melton. Libra: Fair Order-Matching for Electronic Financial Exchanges. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 156–168, 2019.

[37] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[38] Official Journal of the European Union. Official Journal of the European Union - Mifid II Execution Venue Data and Quality of Execution. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0575&from=EN. Accessed: 2021-05-09.

[39] David Schneider. The Microsecond Market. *IEEE spectrum*, 49(6):66–81, 2012.

[40] SEC. Securities and Exchange Commission - Regulation NMS. https://www.sec.gov/rules/final/34-51808.pdf. Accessed: 2021-05-09.

[41] Iain Sheridan. MiFID II in the Context of Financial Technology and Regulatory Technology. *Capital Markets Law Journal*, 12(4):417–427, 2017.

[42] Mao Ye, Chen Yao, and Jiading Gai. The Externalities of High Frequency Trading. *WBS Finance Group Research Paper*, (180), 2013.