POSTER: Jasper, A Scalable and Fair Multicast for Financial Exchanges in the Cloud

Muhammad Haseeb
New York University

Jinkun Geng
Stanford University

Ulysses Butler
New York University

Xiyu Hao
New York University

Daniel Duclos-Cavalcanti
Technical University of Munich

Anirudh Sivaraman
New York University

There has been a growing interest from both industry [8, 11, 12] and academia [7, 9] in migrating financial exchanges to the public cloud because of multiple benefits provided by the cloud, including scalability, robust infrastructure, flexible resource allocation, and potential cost savings [4]. However, migrating a financial exchange from on-prem clusters to the cloud poses several challenges.

Financial Exchanges Requirements: One fundamental requirement for a typical financial exchange is a fair multicast service [7]. Such a service (e.g., NASDAQ's ITCH [10]) is responsible for disseminating information about the state of the market (termed as *market data*) to a large number of market participants (MPs). Market data serves as the important reference for MPs to make trading decisions, and High Frequency Trading (HFT) firms often compete on how quickly they can place trades based on this information. To ensure fairness, market data should be delivered to every MP almost simultaneously, so that no MPs can earn unfair advantages over others. Besides, HFT firms also require consistently low latency from exchange to MPs for market data distribution so that MPs can trade on the most up-to-date information.

Challenges in the Cloud: While a high-performance fair multicast service in on-prem clusters might be implemented using switch support and carefully engineered networks,[1] the situation is much less favorable in the public cloud, where the hardware (e.g., switch) support for *multicast is not usually available* to cloud tenants. The public cloud also exhibits *higher and more varied latency* than on-prem clusters. This variance, combined with the *limited control* a tenant possesses on the underlying network, makes it difficult to realize fair/simultaneous delivery of market data to all the market participants (MPs). As a result, implementing such a multicast service for financial exchanges in the public cloud becomes challenging for cloud tenants.

Jasper: We develop Jasper, an overlay multicast service for cloud-hosted financial exchanges (Figure 1). To achieve low latency while scaling to a large number of receivers, Jasper (1) builds a *tree of proxies* for multicast, instead of having the sender directly unicast its message to all receivers; (2) introduces a *VM hedging* technique to tackle high latency variance; (3) designs a deployment model assuming no trust between MPs and the exchange server that incorporates a *hold-and-release* mechanism [7] (the receivers hold messages and only process them at a deadline) with the *Trusted Execution Environment* (TEE) technique [2] to ensure fairness.

A Proxy Tree: A tree reduces the serialization or transmission delay required to unicast multiple messages back-to-back, but adds additional hops in the sender-to-receiver path. In Jasper, we use the tree structure as a starting point and develop new techniques to lower latency and latency variance and achieve fairness in data delivery while scaling to a large number of receivers. The structure of a proxy tree (depth D and fan-out F) is tuned to minimize latency. Existing cloud-based exchanges [7–9] implement multicast using the direct unicast approach. This may be considered a special case of a tree where D is 1 and F is N. We show that there is value in increasing the D and decreasing the F as the number of receivers (N) grows. We provide a heuristic for tuning D and F which provides good performance: Given N receivers, we fix $F = 10$ and then derive $D = [log_{10}N]$ (round to the nearest integer). We find that more sophisticated learning-based techniques do not outperform our simple heuristic because of the high variation in latency and performance of VMs in the public cloud.

VM hedging: For achieving consistently low latency and decreasing the latency variance spatially, Jasper introduces a technique called VM hedging, motivated by request hedging [6, 13, 15]. In VM hedging, each VM in the proxy tree receives messages from two or more different sources (i.e., parent and one or more aunts), defined by hedging factor H, where the path lengths for all messages are the same. A VM processes and forwards the message copy to children that is received earliest and discards the rest. VM hedging reduces the impact of latency fluctuations, yields much smaller latency variance, and narrows down the window of time in which all the multicast receivers receive a multicast message.

[1]Some financial firms measure wire lengths [14] to achieve simultaneous delivery of market data to all MPs.

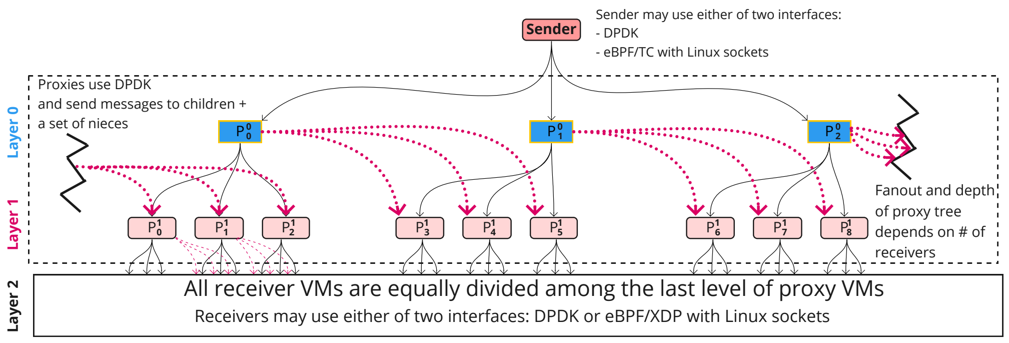
Sender
Sender may use either of two interfaces:
- DPDK
- eBPF/TC with Linux sockets
Proxies use DPDK and send messages to children + a set of nieces
Layer 0
Layer 1
Layer 2
Fanout and depth of proxy tree depends on # of receivers
All receiver VMs are equally divided among the last level of proxy VMs
Receivers may use either of two interfaces: DPDK or eBPF/XDP with Linux sockets

Figure 1: A Jasper Tree. Dotted edges represent hedging, which lowers the latency variance.

Jasper Deployment Model: To ensure perfect fair delivery of data, the exchange requires the MPs to run a hold-and-release mechanism, as proposed by CloudEx [7]. In this mechanism, deadlines are attached to the messages, and an MP is supposed to receive a message and only process it at or after a deadline associated with the message. The deadlines are set, by the exchange in a way to ensure that every MP will receive the message by the deadline. The efficacy of this fair delivery mechanism relies on the MPs to respect the hold-and-release protocol. However, MPs have the incentive to *not respect the protocol and process the messages before the deadline* to gain an advantage over the others. This raises security concerns in Jasper. We assume no trust between MPs and exchanges in Jasper deployment: MPs do not trust their trading programs are exempt from hijacks if running on the VMs provided by the exchange. On the other hand, the exchange does not trust MPs will honestly execute the hold-and-release protocol on the VMs owned by the MPs. To resolve this dilemma, we incorporate the TEE technique, which has become generally available in public cloud [2, 3, 5], to maintain security boundaries between the exchange and traders. More specifically, MPs run their trading algorithm inside the enclaves of the VMs owned by the exchange, and the exchange runs the hold-and-release mechanism outside the enclave. As a result, MPs' trading program is protected by the TEE, and the exchange can also execute the hold-and-release protocol in its owned VMs.

Jasper outperforms AWS Transit Gateway: Figure 2 shows that Jasper distinctly outperforms AWS-TG based multicast and DU (Direct Unicasts approach). The median latency for Jasper is $129\mu s$ where it is $228\mu s$ for AWS TG and $254\mu s$ for DU for 100 multicast receivers. We observe more benefits from Jasper as the number of receivers increases.

VM Hedging reduces delivery window size: VM hedging reduces the spatial variance i.e., messages are received by multicast receivers at almost the same time. Figure 3 shows that delivery window size (the difference between the time when the first receiver and the last receiver receive a multicast message) is reduced with hedging.

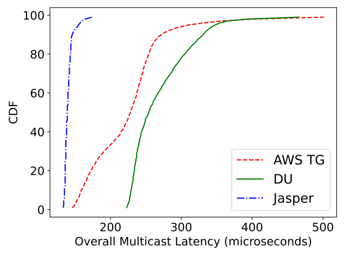

Figure 2: Jasper outperforms DU, and AWS TG.

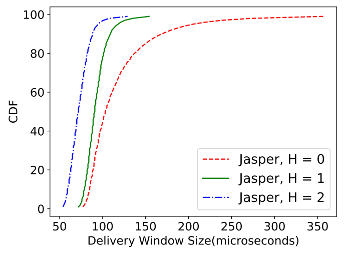

Figure 3: Hedging reduces delivery window size.

Next Steps: Currently Jasper mainly focuses on the performance improvement on the ITCH direction (i.e., market data dissemination from the exchange to MPs). However, the cloud environment also presents challenges in the OUCH direction (i.e., order submission from a large number of MPs to an exchange server). We have several avenues to explore as the next steps:

(1) A **distributed limit order book** [1] would enable the exchange to scale horizontally and further improve the throughput of the system.

(2) A **bucketed integer priority queue**, instead of simple priority queues at the exchange, is more desirable to build high-performance exchange systems, which avoids the overheads of locking the data structures and enables more parallelism of order processing.

(3) **Smartly prioritizing the critical orders** (i.e., orders that are more likely to get executed) when orders are sent from MPs to the exchange server through the proxy tree. Such prioritized order submission would trigger more orders matched and further improve the liquidity of the market.

REFERENCES

[1] Frédéric Abergel, Marouane Anane, Anirban Chakraborti, Aymen Jedidi, and Ioane Muni Toke. 2016. *Limit order books*. Cambridge University Press.

[2] AWS. 2024. AWS Nitro Enclaves. https://aws.amazon.com/ec2/nitro/nitro-enclaves/. Accessed: 2024-05-22.

[3] Azure. 2023. Trusted Execution Environment (TEE). https://learn.microsoft.com/en-us/azure/confidential-computing/trusted-execution-environment. Accessed: 2024-05-22.

[4] Sara Castellanos. [n. d.]. Nasdaq Ramps Up Cloud Move. https://www.wsj.com/articles/nasdaq-ramps-up-cloud-move-11600206624. Accessed: 2024-01-31.

[5] Alibaba Cloud. 2024. Build an SGX confidential computing environment. https://www.alibabacloud.com/help/en/ecs/user-guide/build-an-sgx-encrypted-computing-environment. Accessed: 2024-05-22.

[6] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Commun. ACM* 56 (2013), 74–80. http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

[7] 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 *Proceedings of the Workshop on Hot Topics in Operating Systems* (Ann Arbor, Michigan) *(HotOS '21)*. Association for Computing Machinery, New York, NY, USA, 96–103. https://doi.org/10.1145/3458336.3465278

[8] Junzhi Gong, Yuliang Li, Devdeep Ray, KK Yap, and Nandita Dukkipati. 2024. Octopus: A Fair Packet Delivery Service. *arXiv preprint arXiv:2401.08126* (2024).

[9] Eashan Gupta, Prateesh Goyal, Ilias Marinos, Chenxingyu Zhao, Radhika Mittal, and Ranveer Chandra. 2023. DBO: Fairness for Cloud-Hosted Financial Exchanges. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 550–563. https://doi.org/10.1145/3603269.3604871

[10] NASDAQ. [n. d.]. Nasdaq TotalView-ITCH 5.0. https://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/NQTVITCHSpecification.pdf. Accessed: 2024-02-02.

[11] Nasdaq.com. [n. d.]. Nasdaq and AWS Partner to Transform Capital Markets. https://www.nasdaq.com/press-release/nasdaq-and-aws-partner-to-transform-capital-markets-2021-12-01. Accessed: 2024-01-26.

[12] Alexander Osipovich. [n. d.]. Google Invests 1 Billion in Exchange Giant CME, Strikes Cloud Deal. https://www.wsj.com/articles/google-invests-1-billion-in-exchange-giant-cme-strikes-cloud-deal-11636029900. Accessed: 2021-02-02.

[13] Mia Primorac, Katerina Argyraki, and Edouard Bugnion. 2021. When to Hedge in Interactive Services. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 373–387. https://www.usenix.org/conference/nsdi21/presentation/primorac

[14] Andrew Smith. 2014. Fast money: the battle against the high frequency traders. *The Guardian* 7 (2014).

[15] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. 2013. Low latency via redundancy. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies* (Santa Barbara, California, USA) *(CoNEXT '13)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2535372.2535392