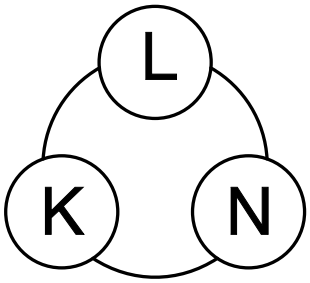

Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Wolfgang Kellerer

Master‘s Thesis

VM Selection Heuristic for Financial Exchanges on the Cloud

Author:	Duclos-Cavalcanti, Daniel
Address:	230 W 55th Street
	10019 New York, NY
	U.S.A.
Matriculation Number:	03692475
Supervisor:	Muhamadd Haseeb & Navidreza Asadi
Begin:	03. April 2024
End:	03. October 2024

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 11.06.2014	
Place, Date	Signature

München, 11.06.2014	
Place, Date	Signature

Kurzfassung

A short abstract of the thesis in German.

Abstract

Financial exchanges consider a migration to the cloud for scalability, robustness, and cost-efficiency. Jasper presents a scalable and fair multicast solution for cloud-based exchanges, addressing the lack of cloud-native mechanisms for such. To achieve this, Jasper employs an overlay multicast tree, leveraging clock synchronization, kernel-bypass techniques, and more. However, there are opportunities for enhancement by confronting the issue of inconsistent VM performance within identical instances. LemonDrop tackles this problem, detecting under-performing VMs in a cluster and selecting a subset of VMs optimized for a given application's latency needs. Yet, we believe that LemonDrop's approach of using time-expensive all-to-all latency measurements and an optimization routine for the framed Quadratic Assignment Problem (QAP) is overly complex. The proposed work aims to develop a simpler and scalable heuristic, that achieves reasonably good results within Jasper's time constraints.

Contents

Chapter 1

Introduction

This chapter should give a short overview over the whole thesis. It should provide background information on the thesis topic, introduce the task definition and give a short outlook on the rest of the thesis.

Chapter 2

Background

In this chapter we discuss the required background on which this work builds upon. We start by discussing a quick overview of financial exchanges, the challenges to their migration to the cloud, cloud computing and the current state-of-the-art in the matter.

2.1 Financial Exchanges

Financial exchanges are a natural evolution of the traditional marketplace concept. By providing a centralized platform to exchange goods, they facilitate the continuous interaction between buyers and sellers. Market participants can engage in bidding, which involves offering to purchase an asset at a specified price, and asking, which refers to the intention to sell an asset at a specified price. This process, known as price discovery, plays a crucial role in determining the fair market value of any asset. A true and fair price benefits both buyers and sellers, since a seller wishes to obtain just compensation, as well as a buyer wishes to pay nothing beyond what is needed.

Modern financial exchanges achieve this by utilizing highly-engineered infrastructures, consisting of on-premise data centers designed to ensure a fair and efficient digital marketplace. These facilities are optimized for low-latency communication, enabling participants to execute trades quickly and reliably, while ensuring that all market participants, co-located within the data center, have equal access to market information and trading opportunities. The key components of a financial exchange are outlined in the following sections and are also illustrated in the diagram in Figure 2.1.

Orders. Trading orders can be either bids or asks. As mentioned previously, bid orders represent intentions to buy an asset at a specified price, while ask orders indicate an offer to sell an asset at a specified price. Another important concept is the limit order book (**LOB**), which is an aggregation of all bids and asks for a specific asset, providing a comprehensive summary of the asset's current market valuation. Figure 2.2 illustrates an example of this data structure as presented to market participants (**MPs**).

Central Exchange Server. At the core of the Central Exchange Server (**CES**) lies a matching engine (**ME**) that receives and processes trading orders from market participants. Orders are sent to the CES, which in return may or may not match an incoming order with a suitable counterpart. Arriving orders, matched orders as well as periodic snapshots of the market (LOBs) are regular events that are broadcasted from the CES to existing MPs.

Market Participants. Market participants are the co-located traders that are directly connected to an exchange's system, issuing orders and receiving data from the exchange.

Gateways. The gateways are structures placed between traders and the CES. Their responsabilities consist of routing orders and market data, as well as protecting the CES from abuse, e.g., unauthenticated or invalid orders. Within the on-premise clusters, gateways are made to be equidistant to the CES as to ensure fairness regarding communication delays between themselves and the central exchange.

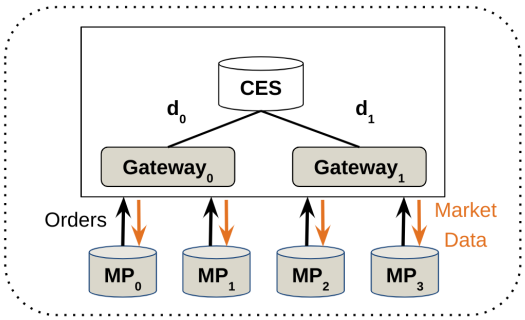

Figure 2.1: On-premise Datacenter

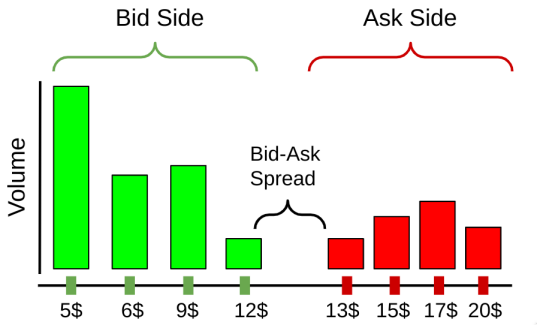

Figure 2.2: Limit Order Book

2.1.1 Migration to the Cloud

Given the demanding network performance of financial exchanges, the use of on-premise data centers is the prevailing standard. However, the cloud displays well-known advantages as a medium for communication. These include flexibility, scalability, robustness and potential cost savings. On the other hand, the public cloud also exhibits high latency variance [HCW+24]. Additionally, it does not provide low-level engineering control to cloud-tenants such as switch-based multicast. As a result, cloud-based systems implement multicasting by directly unicasting a copy of a message to each recipient [GSPR22, GGS+21, GLR+24, GGM+23].

A fundamental characteristic of a financial exchange is the ability to disseminate data to market participants in a fair manner. That is, all MPs receive updates to the market almost simultaneously, as to not allow unwanted arbitrage opportunities. Without native solutions to enable multicasting and the significant serial delay of direct unicasting, a clear problem is presented to achieve the possible migration of financial exchanges to the public cloud.

2.1.2 Jasper: A Scalable Financial Exchange in the Cloud

A siginificant project in this domain is Jasper [HGB+24], which offers a modern implementation of a financial exchange on the public cloud. This work realizes a multicast service that achieves low latency of less than 250 µs and a latency difference of less than 1 µs across 1,000 multicast receivers. To accomplish this, a series of techniques are employed.

Overlay Proxy Tree. This is the starting point for Jasper's design and borrow's the concept of trees to scale communication to a larger number of receivers, while improving latency in comparison to direct unicasting [CEM07, SST09]. The structure of the tree with Depth (**D**) and fan-out (**F**) has been empirically established as seen in Equations [2.1, 2.2].

$$F = 10 \tag{2.1}$$

$$D = \log_{10}(N) \tag{2.2}$$

Clock Synchronization. Jasper utilizes high-precision software clock synchronization to compensate for noisy conditions in the cloud [GLY+18]. Additionally, a common clock across a cluster allows for precise global timestamps to coordinate actions across nodes.

Hold and Release. Even with a proxy tree there still are serial delays that impact every hop in the system. Therefore, receivers located at the leaves of this tree will not receive packets simultaneously. By leveraging global timestamps, a technique called hold and release is applied where a chosen deadline is appended to every packet, instructing receivers when they are allowed to consume the data. That way, a temporal barrier is used to enforce simultanous updates from the market to all recipients. The deadline is set to a future point in time at which all receivers are highly likely to have received the message. To calculate these deadlines, the sender or root of the tree continuosly gathers one-way-delay measurements between itself to all receivers in the tree.

2.2 Cloud Computing

Before personal computers became widespread, they were famously expensive for the everyday consumer. It wasn't until the 1980s and 1990s that this changed. However, building large-scale applications still required significant knowledge in configuring, managing, and maintaining extensive computing resources. In 2006, Amazon introduced Elastic Compute Cloud (EC2), a service that takes upon the responsability of providing and maintaining computational resources, which could be then rented to users per unit-time [Ser06]. EC2 offers units of compute, memory, and storage, all connected through a networking infrastructure allowing communication between these units. Through advancements in hardware virtualization, these resources could be allocated to developers through Virtual Machines (**VMs**) rather than being tied to specific physical instances. This brought forth a paradigm

well known today as cloud computing, allowing for a more flexible use of compute, as machines could be up- and down-scaled more efficiently. Many large corporations such as Google, IBM and Microsoft have followed and today the public cloud is an established medium for application deployment [Clo08, IBM11, Mic10]. Moreover, it is a significant technological industry, generating global spending that exceeds $700 billion dollars per year [Mic21].

Despite the cloud's inherent flexibility, allowing applications to dynamically scale, works have shown that VMs with the same configuration often exhibit significant performance differences [VMW, XMNB13]. This variability is largely attributed to the cloud's multi-tenant nature, where VMs may share CPU, memory, and network resources with other VMs. This resource contention can result in uneven performance, especially when one or more VMs, commonly referred to as *"noisy neighbors"*, consume an excessive amount of shared resources. These underperforming VMs, labeled as *"stragglers"* in the literature, pose a challenge in maintaining predictable and efficient cloud-based computations. Stragglers can significantly delay the overall completion of tasks, particularly in environments where large-scale distributed computing takes place, reducing system throughput, causing inefficiencies, and ultimately leading to higher operational costs.

Research efforts have focused on mitigating the negative impact of straggler VMs in a cluster [YAK14, YHG+17, YHGK15]. These optimizations have been especially critical in the context of batch data processing frameworks, where task completion time can heavily depend on the performance of the slowest VM in the cluster. As cloud infrastructure continues to evolve, ongoing research aims to further enhance the detection and handling of stragglers, ensuring that applications deployed in the cloud remain resilient and performant, even when encountering unpredictable resource contention.

2.2.1 LemonDrop

One notable advancement in the field of optimizing straggler VMs is LemonDrop, designed to select and schedule virtual machines (VMs) based on an application's latency needs [Sac22]. LemonDrop utilizes an accurate clock synchronization algorithm, allowing it to detect fine-grained anomalies in one-way delays (OWD) between pairs of VMs [GLY+18]. This approach allows LemonDrop to uncover subtle performance variations that could otherwise go unnoticed.

What distinguishes LemonDrop is its ability to frame the selection and scheduling process as a natural Quadratic Assignment Problem (QAP) [KB57]. This allows an efficient assignment and scheduleing of the top-performing K VMs from a pool of $N > K$, optimizing both latency and overall performance. Unlike prior works that primarily focused on mitigating stragglers in batch processing, LemonDrop is particularly tailored to latency-sensitive applications, making it highly effective in real-time, interactive cloud environments. Furthermore, the system's ability to detect underperforming VMs makes it valuable for autoscaling and maintaining application responsiveness.

Algorithm 1 FAQ Algorithm

function FAQ(Λ, Δ, ϵ, n)
  $P^{(0)} = 11^T$
  $i = 0$
  $s = 0$
  while $s = 0$ **do**
    $\nabla f(P^{(i)}) = -\Lambda P^{(i)} \Delta^T - \Lambda^T P^{(i)} \Delta$ ▷ Gradient wrt current solution
    $Q^{(i)} = \arg\min_{P \in \mathcal{D}} \operatorname{trace}(\nabla f(P^{(i)})^T P)$ ▷ Direction which minimizes 1st order approx of $f(P)$
    $\alpha^i = \min_{\alpha \in [0,1]} f(P^{(i)} + \alpha Q^{(i)})$ ▷ Best step size in chosen direction
    $P^{(i+1)} = P^{(i)} + \alpha^i Q^{(i)}$ ▷ Update our current solution
    if $\|P^{(i)} - P^{(i-1)}\|_F < \epsilon$ **then** $s = 1$ ▷ Stop opt
    if $i > n$ **then** $s = 1$ ▷ Stop opt
    $i = i + 1$ ▷ Iteration number
  end while
  $P_{\text{final}} = \arg\min_{P \in \mathcal{P}} P^{(i-1)} P^T$
  return P_{final}
end function

2.3 Motivation

Chapter 3

Implementation/Results

3.1 Implementation

At the core of the tool.

3.2 Results

Results of the performed investigations are presented here. Interpretations for the observed effects are given and the impact of investigations is discussed.

Chapter 4

Conclusions and Outlook

The thesis is concluded here. The considered problem is repeated. The contribution of this work is highlighted and the results are recapitulated. Remaining questions are stated and ideas for future work are expressed.

Appendix A

Artifacts

The appendix may contain some listings of source code that has been used for simulations, extensive proofs or any other things that are strongly related to the thesis but not of immediate interest to the reader.

Appendix B

Abbreviations

CES	Central Exchange Server
MP	Market Participant
VM	Virtual Machine

Bibliography

[CEM07] Tatsuhiro Chiba, Toshio Endo, and Satoshi Matsuoka. High-performance mpi broadcast algorithm for grid environments utilizing multi-lane nics. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, pages 487–494, 2007.

[Clo08] Google Cloud. Google cloud platform. `https://cloud.google.com/`, 2008. Accessed: 2024-09-12.

[GGM+23] Prateesh Goyal, Eashan Gupta, Ilias Marinos, Chenxingyu Zhao, Radhika Mittal, and Ranveer Chandra. Dbo: Response time fairness for cloud-hosted financial exchanges, 2023.

[GGS+21] Ahmad Ghalayini, Jinkun Geng, Vighnesh Sachidananda, Vinay Sriram, Yilong Geng, Balaji Prabhakar, Mendel Rosenblum, and Anirudh Sivaraman. Cloudex: a fair-access financial exchange in the cloud. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '21, page 96–103, New York, NY, USA, 2021. Association for Computing Machinery.

[GLR+24] Junzhi Gong, Yuliang Li, Devdeep Ray, KK Yap, and Nandita Dukkipati. Octopus: A fair packet delivery service, 2024.

[GLY+18] 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, page 81–94, USA, 2018. USENIX Association.

[GSPR22] Jinkun Geng, Anirudh Sivaraman, Balaji Prabhakar, and Mendel Rosenblum. Nezha: Deployable and high-performance consensus using synchronized clocks. *Proceedings of the VLDB Endowment*, 16(4):629–642, December 2022.

[HCW+24] Owen Hilyard, Bocheng Cui, Marielle Webster, Abishek Bangalore Muralikrishna, and Aleksey Charapko. Cloudy forecast: How predictable is communication latency in the cloud? In *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2024.

[HGB+24] Muhammad Haseeb, Jinkun Geng, Ulysses Butler, Xiyu Hao, Daniel Duclos-Cavalcanti, and Anirudh Sivaraman. Jasper: Scalable and fair multicast for financial exchanges in the cloud, 2024.

[IBM11] IBM. Ibm cloud. `https://www.ibm.com/cloud/`, 2011. Accessed: 2024-09-12.

[KB57] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

[Mic10] Microsoft. Microsoft azure: Cloud computing services. `https://azure.microsoft.com/`, 2010. Accessed: 2024-09-12.

[Mic21] Microsoft. Microsoft azure: Cloud computing services. `https://www.idc.com/getdoc.jsp?containerId=prUS48208321`, 2021. Accessed: 2022-12-05.

[Sac22] Vighnesh Sachidananda. *Scheduling and Autoscaling Methods for Low Latency Applications*. Stanford University, 2022.

[Ser06] Amazon Web Services. Amazon ec2: Elastic compute cloud. `https://aws.amazon.com/ec2/`, 2006. Accessed: 2024-09-12.

[SST09] Peter Sanders, Jochen Speck, and Jesper Larsson Träff. Two-tree algorithms for full bandwidth broadcast, reduction and scan. *Parallel Comput.*, 35(12):581–594, dec 2009.

[VMW] VMWare. Vmware - performance tuning with numa. `https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-1DAB8F35-BA86-4063-8459-55D2979B593E.html`. Accessed: 2024-09-12.

[XMNB13] Yunjing Xu, Zachary Musgrave, Brian Noble, and Michael Bailey. Bobtail: Avoiding long tails in the cloud. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 329–341, Lombard, IL, April 2013. USENIX Association.

[YAK14] Neeraja J Yadwadkar, Ganesh Ananthanarayanan, and Randy Katz. Wrangler: Predictable and faster jobs using fewer resources. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14, 2014.

[YHG+17] Neeraja J. Yadwadkar, Bharath Hariharan, Joseph E. Gonzalez, Burton Smith, and Randy H. Katz. Selecting the best vm across multiple public clouds: a data-driven performance modeling approach. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, page 452–465, New York, NY, USA, 2017. Association for Computing Machinery.

[YHGK15] Neeraja Jayant Yadwadkar, Bharath Hariharan, Joseph E. Gonzalez, and Randy H. Katz. Faster jobs in distributed data processing using multi-task learning. In *SDM*, 2015.