UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANNA BEATRIZ LUCENA LIRA

SEM CONFLITO NO CACHE: OBSERVAÇÕES DE UMA PLATAFORMA DE COMÉRCIO ELETÔNICO MULTI-TENANT

CAMPINA GRANDE - PB

2023

ANNA BEATRIZ LUCENA LIRA

SEM CONFLITO NO CACHE: OBSERVAÇÕES DE UMA PLATAFORMA DE COMÉRCIO ELETÔNICO MULTI-TENANT

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Thiago Emmanuel Pereira da Cunha Silva

CAMPINA GRANDE - PB

2023

ANNA BEATRIZ LUCENA LIRA

SEM CONFLITO NO CACHE: OBSERVAÇÕES DE UMA PLATAFORMA DE COMÉRCIO ELETÔNICO MULTI-TENANT

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Thiago Emmanuel Pereira da Cunha Silva

Orientador – UASC/CEEI/UFCG

Reinaldo Cezar de Morais Gomes

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 17 de NOVEMBRO de 2023.

CAMPINA GRANDE - PB

RESUMO

O armazenamento em cache é uma técnica clássica para aumentar o desempenho do sistema, reduzindo a latência percebida pelo cliente e a carga do servidor. No entanto, projetar e configurar cuidadosamente o cache é uma tarefa desafiaforal. Inclui a escolha do tamanho da capacidade, da política de remoção de itens, entre outros aspectos. Esta tarefa é ainda mais complexa em sistemas multi-inquilinos, nos quais cada inquilino opera de forma independente e apresenta demandas diferentes ao longo do tempo. Por essa razão, a configuração eficaz e a gestão do cache requerem uma compreensão das características da carga de trabalho, incluindo níveis de carga, padrões de acesso e localidade temporal. Este artigo concentra-se na caracterização da carga de trabalho de um cache multi-inquilino em uma grande plataforma de comércio eletrônico. Encontramos uma diversidade significativa entre inquilinos em relação a padrões de carga, utilidade do cache e localidade temporal. Com base nisso, destacamos estratégias para otimizar a gestão de sistemas de cache multi-inquilino, adotando políticas de admissão e ajuste dinâmico de capacidade (escalonamento). A implementação dessas estratégias em um ambiente de produção constituirá uma fase subsequente desta investigação.

NO CLASH ON CACHE: OBSERVATIONS FROM A MULTI-TENANT ECOMMERCE PLATFORM

ABSTRACT

Caching is a classic technique for boosting system performance by reducing client-perceived latency and server load. However, carefully designing and configuring the cache is a challenging task. It includes choosing capacity size and eviction policy, among others. This task is more challenging in multi-tenant systems, where each tenant operates independently and exhibits different demands over time. For this reason, effective cache configuration and management require an understanding of workload characteristics, including load levels, patterns of access, and temporal locality. This paper focuses on the workload characterization of a multi-tenant cache of a large ecommerce platform. We find a significant diversity of tenants regarding load patterns, cache usability, and temporal locality. Based on this, we highlight strategies to optimize the management of multi-tenant cache systems by adopting admission policies and dynamic capacity adjustment (scaling). Implementing these strategies within a production environment will constitute a subsequent phase in this investigation.

No clash on cache: observations from a multi-tenant ecommerce platform

Anna Beatriz Lucena Lira
anna.lira@ccc.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraíba, Brazil

Thiago Emmanuel Pereira da Cunha Silva
temmanuel@computacao.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraíba, Brazil

ABSTRACT

Caching is a classic technique for boosting system performance by reducing client-perceived latency and server load. However, carefully designing and configuring the cache is a challenging task. It includes choosing capacity size and eviction policy, among others. This task is more challenging in multi-tenant systems, where each tenant operates independently and exhibits different demands over time. For this reason, effective cache configuration and management require an understanding of workload characteristics, including load levels, patterns of access, and temporal locality. This paper focuses on the workload characterization of a multi-tenant cache of a large ecommerce platform. We find a significant diversity of tenants regarding load patterns, cache usability, and temporal locality. Based on this, we highlight strategies to optimize the management of multi-tenant cache systems by adopting admission policies and dynamic capacity adjustment (scaling). Implementing these strategies within a production environment will constitute a subsequent phase in this investigation.

KEYWORDS

Web Service. Cache. Workload. Tenant.

1 INTRODUCTION

Caching is a classic that never dies. From computer organization to cloud-based web systems, caching reduces client-perceived latency and service load.

To fulfill its potential as a performance booster, cache systems need to be carefully designed and configured. This includes choosing parameters, such as cache capacity size and eviction algorithms. Failures in configuration lead to direct impacts on the quality of service (usually observed in miss/hit ratio indicators) or resource waste (i.e., when the cache capacity is over-provisioned and additional capacity does not improve performance).

Effective cache configuration requires understanding workload characteristics. A typical starting point is to estimate the load level. The number of servers used in a large cache service depends on this load-level information.

However applicable, knowing load levels alone cannot define other cache parameters, such as capacity. This is because many requests sent to the cache might be related to a few cached items, thus reducing the need for more cache capacity. Also, it is well-known that cache usage is made in phases, and most of the operations in a phase are related to a subset of the cached items (the working set). All these temporal locality aspects affect cache capacity rightsizing.

Practitioners are well aware of the importance of considering this advice. However, one factor deviates practice from good practice: multi-tenancy.

To illustrate, consider multi-tenant ecommerce platforms, the case study of this paper. In these platforms, each tenant is an enterprise independent from the others; each tenant has its clients and products. However independent, the tenant's ecommerce sites run on shared resources and services (including caches) owned and managed by the platform. Managing multi-tenant caches is more complex than managing single-tenant ones because all the tenants are unlikely to behave equally. And, as we described, cache services are sensitive to load characteristics. There might be tenants that sell more than others. There must be tenants with large and small product inventories. There must be tenants with seasonal and sporadic selling patterns. All these characteristics affect cache management.

To uncover these factors and highlight the challenges of multi-tenant caching, we collected and analyzed one of the web cache services from a large-scale ecommmerce platform[1]. The observed cache service supports a few thousand tenants from different time zones for a 10 hours observation period.

We found that diversity is the norm. Aggregated loads vary up to three times, reaching more than 150,000 requests per minute. Also, aggregate load follows the same overall platform traffic trend (high traffic until late at night and lower traffic at dawn). However, we have a completely different picture when we analyze tenants in isolation. Some tenants exhibit stable load, others show a periodic load pattern, while some show peak periods (even at unexpected hours). Also, the load is highly concentrated among tenants: the 10% more loaded tenants account for almost 80% of the aggregated load.

We also analyzed cache friendliness. A cache-friendly workload shows a high repetition degree of access to cache items; this is good for cache because the higher the repetition, the higher the chances of cache hits (mainly when repeated accesses occur in a short interval). At one side of the spectrum, a workload with no repetition leads only to cache misses. We found that many tenants exhibit low repetition. Half of the tenants show up to 40%; a direct implication is that these tenants cannot have more than the 40% of hit ratio, regardless of the cache configuration and capacity. Repetition also varies over time. While some tenants sustain low or high repetition during the observed period, others change their behavior as time passes.

Trabalho de Conclusão de Curso, Bacharelado em Ciência da Computação, 2023
.

[1] https://vtex.com

We found diversity again when we analyzed temporal locality. While the top-1 most loaded (in the full trace) tenant has 5.1% of requests made in a selected minute, the top-2 tenant has 0.1% requests made on a same minute interval. In addition, there is a high variation in temporal locality across time. For example, across the 10-hour observation, the number of requests for top-1 tenant goes from 0.36 to 1 (normalized).

The remainder of this paper is organized as follows. In Section 2, we describe the procedure for collecting data from the ecommerce platform in production. Section 3 describes the context and metrics applied in the workload characterization presented in section 4. In addition, Section 5 describes the implications of the observed workload characteristics for cache management. Section 6 presents the conclusions and future work of this research. Acknowledgments are presented in the Section 7. Finally, the references.

2 WORKLOAD

This section describes the workload. Section 2.1 overviews the system that the studied cache service is in, giving a brief description of what kind of data is managed. Section 2.2 describes the non-intrusive instrumentation for data collection on production servers. Also, Section 2.3 explains the data used in the analysis, giving details about what compounds.

2.1 System Overview

The data was collected from a prominent Brazilian ecommerce business-to-business provider. This provider offers tools and services to support other international ecommerce enterprises in creating and operating their online stores.

The system within which the cache service examined in this research operates is responsible for the management of product data for numerous catalog enterprises. The macro system is responsible for product management, comprehending from pricing to product data storage. The system that generates the data that we studied in this work is responsible for product data management, and we can call it a Catalog system. Figure 1 shows an overview of the system. Each stored product has associated data, including text fields (e.g. titles, descriptions, product identifier, and enterprise identifier), as well as references to the product images. The Catalog system comprises Load Balancers and services responsible for managing these products. The database service that serves this system is a type of index for the product data.

The cache service under investigation is implemented as a cluster of NGINX[1] servers, and its primary role is to store the response data generated by all the systems (e.g. Catalog) within the aforementioned macro system that communicates with the database service. Each response stored in this service contains the requested product data. It utilizes a key-value format. A key is generated by applying a hash function to the URI that uniquely identifies a request. The stored value is the retrieved product information.

2.2 Data Collection

As mentioned in Section 2.1, the cache service under examination is responsible for managing response data generated by all the services within the system previously described.

The collected data consists of logs generated by requests for product data from various enterprises. Each request can be redirected to different services of the macro system. As previously mentioned, the data collection focuses on data generated from requests to the Catalog system.

The typical path these requests follow is that they are initially routed to the APIs, passing by the cache service before querying the database. The response data generated from these database query requests are stored in the cache, and the NGINX nodes produce corresponding log entries when this data is captured and sent to a Search Engine.

The log entries encompass various fields that provide information about the requested product data and describe the request parameters. These fields include request details such as the URI, among others. Additionally, the log entries contain information related to the cache service itself, which includes details like the cache status, the specific NGINX node that responded to the request, the timestamp, and the time wait.

The Search Engine under consideration operates within a high-traffic production environment, where data collection must be conducted without causing disruption or performance degradation. To achieve this, data was collected using a 10% sample of all logs. This sampling process was achieved by collecting a continuous 10% slice of the logs within each 5-minute interval.

The data collection process was initiated by executing a script within the same network as the Search Engine. The sampling parameters were configured directly within this script. The script collected sampled data at 5-minute intervals over ten hours, specifically from 21:00 to 7:00 GMT-3. The script collects a 5-minute interval log, compresses it, and sends it to Cloud Storage. Figure 2 presents a brief overview of how this collection was executed.

2.3 About the Data

In the collected workload time interval, it was possible to get 60.9 million request log entries, signifying a substantial volume of requests and cache service information. As previously mentioned, a request signifies a query for product data associated with a particular tenant. So, these logs correspond to requests for approximately 25 million different product data belonging to approximately five thousand businesses. Notably, each tenant exhibits unique and significant access patterns, which section 4 will explore further.

Figure 3 presents the normalized number of incoming requests received per minute throughout the data collection period, illustrating the variations and trends in request rates.

As aforementioned, each entry collected during the data collection process includes information regarding the requested data and details about the cache service. To streamline and facilitate our analysis, we have chosen to focus on three key fields which are critical for our research. These fields include tenant identification, product identification, and the request timestamp. Table 1 shows the data description. Note that, in this work, product and item are considered the same.

3 BACKGROUND

Caching systems for multi-tenant ecommerce platforms present configuration and management challenges. This is because each

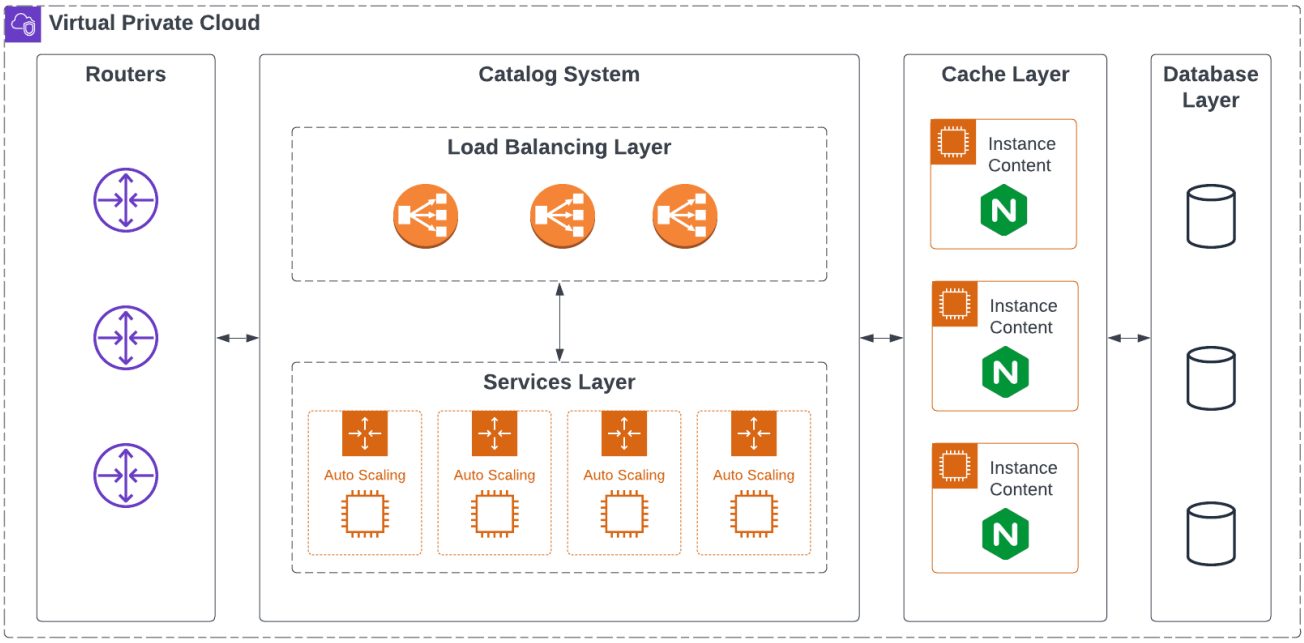

Figure 1: Simplified architecture of Catalog system developed by a big ecommerce provider and how it uses the studied cache layer.

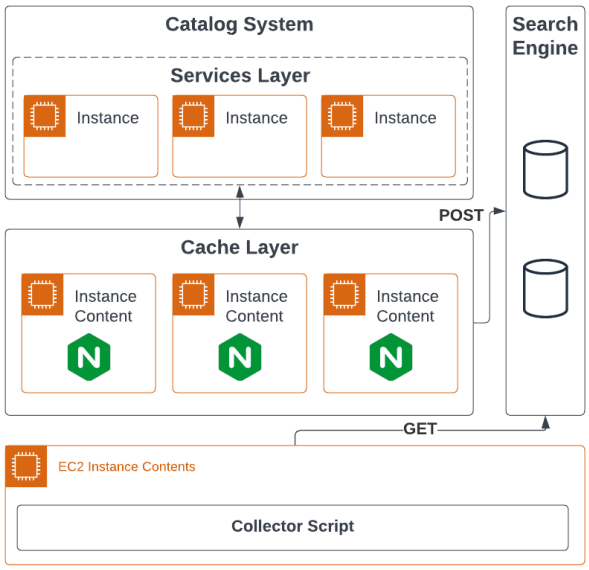

Figure 2: Diagram describing the communication between the cache service, search engine, and the collector script.

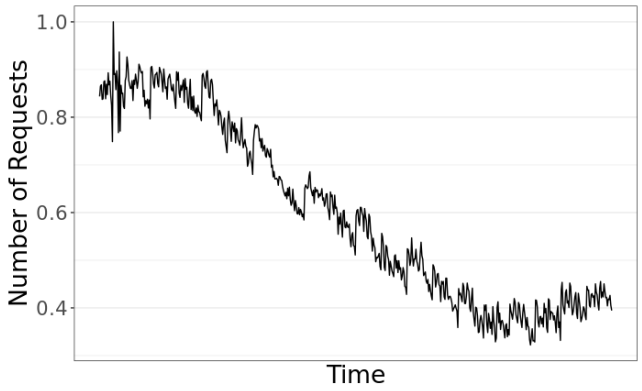

Figure 3: Normalized incoming request rates per minute, highlighting request rate variations and trends over the data collection period.

Table 1: Selected fields used in this work.

Column	Description
`Timestamp`	Time instant that the request was sent
`Tenant ID`	Hashed enterprise identifier which offers the item
`Item ID`	Hashed requested item identified

tenant has different peculiarities in its trading, considering that each tenant is an independently operating enterprise. For example, there is variation among tenants in terms of request volume, product inventory, and selling patterns. This results in a system operating at a high load level.

As a result of these different behaviors, it is expected to generalize system configurations and management because it is a complex task to observe the particularities of each tenant. For this reason, there is high resource usage in this type of system, and this often leads to over-provisioning. For example, an overestimated capacity supports the demand of all the tenants and also supports peak demand from tenants at specific times. This leads to under-utilization of resources at times of low load level. Keeping resources idle is a costly practice that does not bring benefits because they are allocated and do not contribute to the quality of the service. Therefore, the need arises to configure the cache effectively.

Careful design and configuration Web caching services require steps beyond the architecture shown in Figure 1. This involves defining the cache capacity, the item expiration time, the eviction policy, and the admission policy. Once configured, one way of evaluating the Cache Quality of Service is to look at the Hit Ratio, defined as $\frac{|HITS|}{|R|}$, where HITS corresponds to the multiset of requests with Hit status and R corresponds to the multiset of all requests.

For an efficient configuration, we need to know the cache demand. However, measuring the demand for cache resources is not a simple task due to several factors. Firstly, requests to the system can vary widely in terms of frequency. In addition, access patterns can change over time due to various factors, such as changes in user behavior. Seasonality also plays a role, with some data being more in demand at specific times. For auxiliary cache configuration, some aspects can be observed to understand the workload necessities. For example, it is important to look for:

Request Arrival The Request Arrival refers to all requests for items arriving in the cache system.

Concentration Of Access Refers to the aggregate counts of accesses to an item.

Footprint The Footprint metric serves as a measure of the data accessed within a specified time window [8]. It is quantified by counting the number of distinct requested items in a time window. If we denote T as a given time window and F as the set of items requested, the footprint is defined as $\text{Footprint}(T) = |F|$. This metric is essential for analyzing system capacity utilization, as it allows us to see how many items are requested in a given period. It also shares a close relationship with the working set size (WSS) theory, which helps to understand an application's memory requirements.

Item Repetition Ratio The Item Repetition Ratio (IRR) stands as a metric for evaluating temporal data access patterns within an application in a time window [4]. Mathematically, IRR is defined as the quotient of the multiset of repeated requests for items (P) and the total of requests for items (R) within a specified time window, expressed as $\text{IRR} = \frac{|P|}{|R|}$. The IRR is an upper bound for the cache hit ratio within a practical caching system over a designated temporal scope. This implies that achieving an IRR-based cache hit ratio signifies an optimal level of cache performance, given the observed access patterns.

Temporal Locality Temporal locality refers to the tendency of the same document to be referenced frequently within short intervals. It differs from concentration, which refers to the aggregate reference counts for items, regardless of the referencing order. Some metrics can be used to measure temporal locality, for example, the LRU stack-depth and Inter-Reference Time. In this work, we will focus on Inter-Reference Time.

Inter-Reference Time Inter-Reference Time represents the time between references to the same item [2]. We calculated the mean of the Inter-Reference Time and called it of MIRT. MIRT sets itself apart from LRU stack-depth analysis by focusing on the timing of data access rather than the order within a stack. While LRU stack-depth provides insight into the sequence in which a specific number of requests occurred within a particular interval, MIRT's primary objective is to define and analyze these time intervals.

Another aspect of cache configuration is the item expiration time. This determines how long the data remains valid before it needs to be updated because the company can change its inventory. In our case, this involves a business rule, so we won't go into ways of better configuring this expiry time.

If only existing admission is on the cache, we need an infinite capacity on the system because the tenant's inventory can be increased, so the footprint can grow infinitely. To control that, cache systems use an eviction policy. When the cache system is entirely fulfilled, these algorithms select which item must be removed to a new item to be cached. Some algorithms are related in literature, such as FIFO, ARC [6], LARC[5] LRU, and Others. For the cache studied in this research, the LRU policy is used, which is the most widely used policy in the literature and real systems. Using the LRU policy, each time an item is accessed, it is marked as the most recently used. When the cache is full, and a new item needs to be stored, the LRU eviction algorithm removes the item that was least recently used.

Another form to improve the cache system is to adopt an admission control policy. These strategies are complementary to the replacement ones. Instead of deciding which items must leave the cache when it is full, an admission control policy decides whether an item should enter the cache. An effective admission policy helps ensure that only the most relevant and frequently accessed data is cached, optimizing the use of these resources. By default, the examined cache does not implement admission policies, presenting an opportunity for enhancing the system's performance. Consequently, we will assess the load to discern potential strategies conducive to implementing an admission policy within this cache.

The following section will present analyses that address the workload of the multi-tenant caching system for the data trace collected. This will be important for identifying ways to improve the configuration and management of the system. It is important to note that the metrics presented in Section 4 were normalized using a min-max function.

4 WORKLOAD CHARACTERIZATION

This section presents a detailed analysis of the referencing behavior of items in multi-tenant caching systems.

Section 4.1 focuses on observing the demand for requests the caching system receives. It shows that the demand for web services varies throughout the day, with peaks after working hours and

again in the early morning hours. It also describes the disparity in the number of requests received per tenant.

Section 4.2 explores the irregular popularity of the items stored in the system, identifying "hot" and infrequently accessed items. The crucial observation is that 80% of all requests are directed at only half of the distinct items in the system.

Subsection 4.3 highlights footprint as an essential metric for measuring the volume of data accessed in a specific time interval. In addition, it reveals temporal variations in footprint as well as variations in footprint per tenant.

Subsection 4.4 presents IRR as a way of evaluating item access patterns. The main point is that IRR shows temporal fluctuations, emphasizing the need for periodic evaluation and adaptive cache management strategies. The IRR also varies between tenants.

Subsection 4.5 evaluates the temporal locality - the frequency with which a document is accessed in short time intervals - of the data by analyzing the Inter-Reference Time.

4.1 Request Arrival

The initial analysis centers on the volume of requests entering the system over ten hours, as detailed in Table 2. The table provides statistical metrics derived from the load, normalized using min-max normalization (ranging from 0 to 1). In this normalized scale, an activity level of 1 signifies the highest activity level observed, while 0 represents the lowest. The findings reveal a mean of 0.41 and a median of 0.37, indicating that both metrics deviate significantly from the maximum value. This suggests a distribution pattern wherein a substantial proportion of observations exhibit relatively low activity levels compared to the peak request rate. The peak of request occurs in the evening, after working hours, it is common for demand for web services to be high. While there is a tendency for demand to fall at daybreak, demand for services rises again in the early hours of the morning [7].

In addition to this trends observed, we can identify seasonality in the number of requests per minute. This refers to patterns that repeat at fixed minute intervals, which may be related to cyclical behavior in a business. We also find random fluctuations or variations that the trend or seasonality cannot explain.

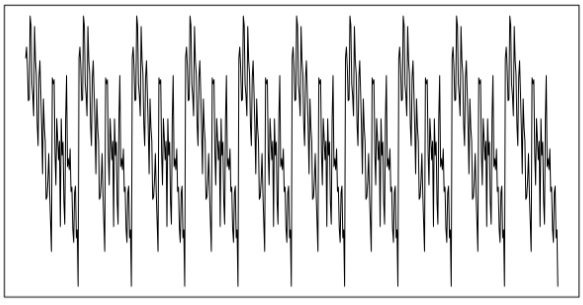

Figure 4: Seasonality found in the load by decomposition of additive time series. Recurring patterns over minutes are probably linked to business cycles.

Table 2: Statistical measures of the number of requests per minute using min-max normalization.

Mean	Median	Max	Min	Std. Dev
0.41	0.37	1	0	0.20

In addition to variation in the number of requests over time for the whole load, a notable disparity in tenant requests becomes evident. A mere 10% of tenants account for approximately 78.42% of all recorded requests. Figure 5 illustrates the cumulative request count for each tenant, ranked accordingly. This visual representation shows the significant disparity in tenant request distribution.

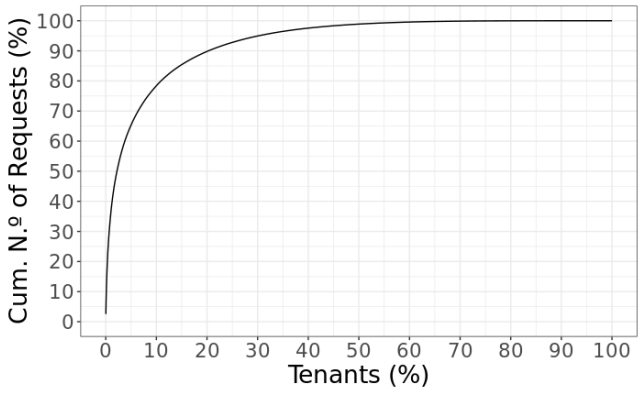

Figure 5: The cumulative request distribution, ranked by tenant. There is a significant disparity in the distribution of tenant requests.

Indeed, the request quantity for each tenant also displays temporal variability. This introduces a layer of complexity in resource allocation. The dynamic nature of request patterns over time means that a tenant's demand for resources can fluctuate, potentially leading to periods of increased demand or sudden spikes in traffic. This may indicate that system management could be more dynamic, sensitive to tenants' needs, and avoiding overprovisioning or underprovisioning. Figure 6 illustrates, for a sample of tenants, the variation in the number of requests over time. Some tenants exhibit peaks in requests, indicating periods of significantly higher demand than their usual levels. This may result from seasonal events, special promotions, or product launches.

Conversely, some tenants maintain relatively constant activity over time, with no significant fluctuations in request quantity. This suggests a more stable and predictable traffic profile associated with a regular customer base. Furthermore, some tenants may display intermittent request patterns, alternating between intense activity periods and relatively calm moments. This oscillation may be influenced by peak shopping times or specific marketing actions. Understanding these diverse behaviors is essential for an efficient allocation of resources. Tenants with request peaks may require more substantial cache allocations during these periods of high demand. Conversely, tenants with more stable request patterns may benefit from more conservative allocations.

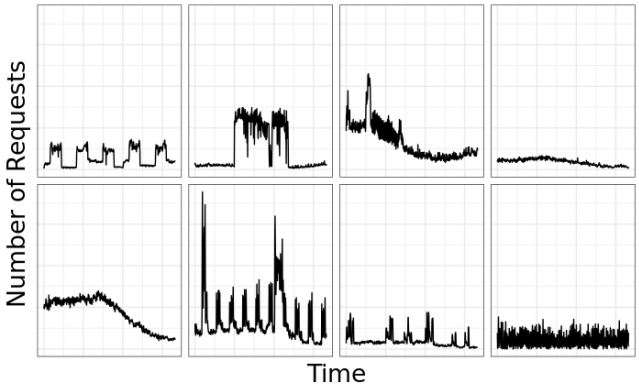

Figure 6: The request variation over time for a selected sample of the biggest tenant by number of requests. Some show request peaks, suggesting periods of increased demand, possibly due to seasonal events, promotions, or product launches.

In order to quantify this variable behavior of the tenants, we calculated the standard deviation of the number of requests by tenants.

When examining the standard deviation between tenants per minute, we observe a low variability in requests within a one-minute interval. Using the K-Means clustering algorithm, we grouped the standard deviations of the tenants into three groups (Low, Medium, and high variability). Figure 7 shows the count of tenants per group. Most tenants show low variability, and some can be considered outliers in group 3. In general, tenants with a high standard deviation may need more proactive support to deal with variations in demand. For tenants with low variability in the data, we can use a more conservative allocation of resources.

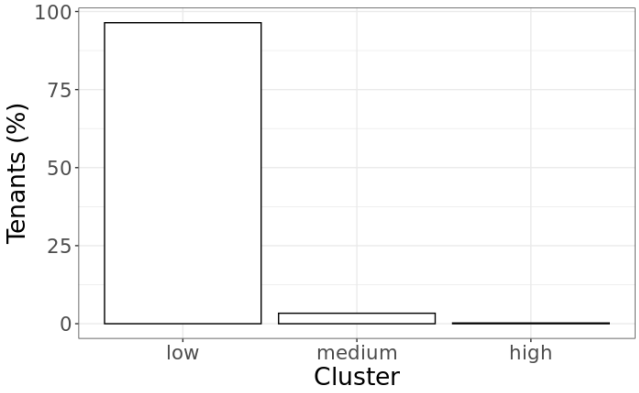

Figure 7: Number of tenants in each group. Many tenants have a low standard deviation, indicating consistency in the request pattern in relation to the average.

4.2 Concentration Of Access

Looking for requested items and evaluating the frequency of access, we can see a significant discrepancy in popularity. Some can be classified as "hot" items with a high access volume. These items are characterized by their great popularity and constant demand. In contrast, other items are rarely accessed and, in some cases, not accessed.

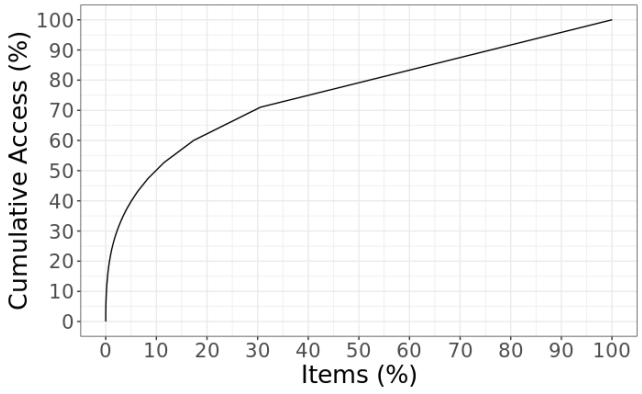

Figure 8: Cumulative frequency of access by item. 80% of all requests are directed at only half of the distinct items.

Figure 8 shows the non-uniform pattern in item referencing behavior. A substantial majority, precisely 80% of all requests, are directed at only half of the distinct items in the system. This phenomenon suggests the existence of a subset of items that are highly requested, while other items show considerably less repetition. This concentration phenomenon is a common characteristic in weblogs, Braun and Claffy [3] have reported in an earlier study.

This implies that not all items warrant being cached. Items with low repetition, i.e., those requested infrequently, may not justify allocating precious cache resources. By retaining these items in the cache, valuable resources may be directed to elements that do not provide a commensurate benefit regarding system efficiency. In this case, an admission control policy could be a good strategy for deciding whether an item should enter the cache.

4.3 Footprint

As mentioned in previous sections, the footprint value depends on the size of the window range chosen. For our trace, with a window T of 10 hours and F the set of items requested in this interval, the footprint is approximately 25 million items.

When we look at the footprint per tenant, with T of 10 hours and F of the set of tenant's requested items, we see that many tenants have a low footprint while others have a large one. Figure 9 illustrates the distribution of Footprints among tenants for the T of 10 hours, highlighting a significant concentration of lower Footprint values. In summary, the 75% of tenants with the smallest footprint represent a portion of only around 10% of the total footprint. Furthermore, Figure 9 also shows tenants with a large Footprint. These tenants can take up a large slice of cached storage.

Also, we can correlate the number of requests and footprint. The Pearson correlation coefficient between the number of requests and footprint is 0.88. This indicates a strong positive correlation. This suggests that, in general, as the number of requests increases, the footprint also tends to increase and is very close to the value 1,

suggesting that this linear association is strong. Correlation does not imply causality. Even if there is a strong correlation between the number of requests and the footprint, we cannot conclude that one causes the other.

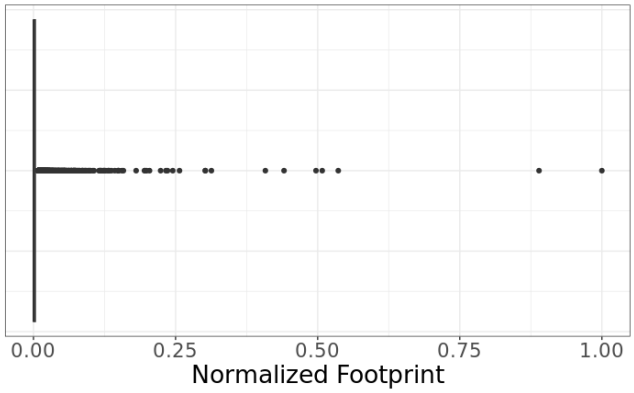

Figure 9: Normalized Distribution footprint per tenant. Some tenants show a big footprint. These tenants can occupy a big slice of the cache storage.

The tenant's specific footprint evolves over time, influenced by their different workload patterns, such as changes in the number of orders. As mentioned, variations in T can lead to changes in the Footprint value. Figure 10 further illustrates this dynamic, showing the variations in footprint per hour for selecting the six largest tenants. This emphasizes that a tenant's footprint can change over time.

To measure the variation in tenant footprint per window, we calculated the standard deviation of the footprint and grouped it by K-Means Algorithm. Figure 11 shows it. Most of the tenants showed low variation. However, a notable observation is the considerable disparity between the maximum and average values of deviation. The maximum is one hundred and four times higher than the average. This substantial difference suggests the presence of atypical tenants characterized by significant footprint variations. In addition, the minimum standard deviation value of 0 is expected since some tenants may have only a single product or only one order in the collected workload, resulting in no variability in their footprints.

Approximately 15% of shopkeepers have standard deviations higher than the average. This finding implies that some tenants' footprints show substantial variations over time, contributing to the higher standard deviation values. These tenants probably have distinct and fluctuating data access patterns that significantly affect their Footprint metrics. Observing the footprint in time windows can help us understand the storage space requirements of each tenant in the system.

4.4 Item Repetition Ratio

A notable observation is the variation in IRR values across different tenants. This implies that sure tenants are more adept at taking advantage of the benefits of caching, while others may not exhibit behaviors favorable to caching. Figure 12 shows many tenants

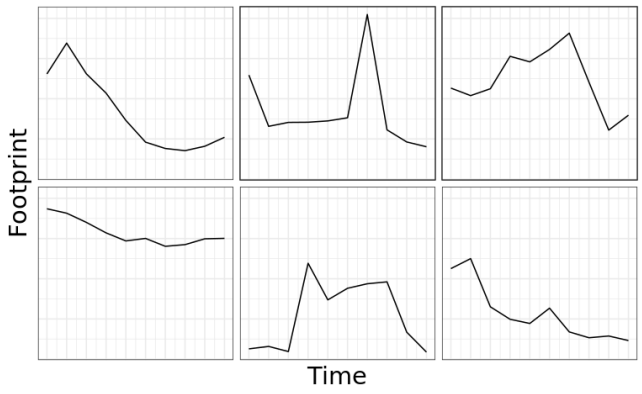

Figure 10: Footprint over time for a selected sample of the biggest tenant by number of requests. Underscores that a tenant's Footprint can change over time.

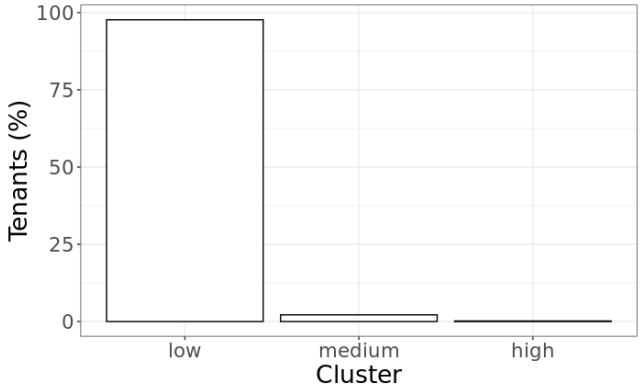

Figure 11: Percentage of tenants by a group based on footprint standard deviation, using the K-means algorithm. This grouping distribution reinforces the correlation between footprint and number of requests.

have low IRR. For example, around 50% of tenants have IRR less than or equal to 40%. With low repetition, these tenants could be candidates for not having items in the cache since their requests in a time window are for different items, not contributing to the cache hit.

By decreasing the window grain for adjacent 1-hour windows, we observe some IRR behaviors over time, as shown in figure 13. Some tenants show high IRR with slight variation, others low IRR with little variation, and others show considerable variation in IRR between windows. We calculated the standard deviation of the tenants' IRR over the windows to quantify these variations. Figure 14 shows the percentage of tenants by a group based on standard deviation, using the K-means algorithm. We observe that, for most tenants, the standard deviation of the IRR is relatively low, which may indicate that tenants have similar access patterns in adjacent windows. This shows that, before entering the cache system, analyzing the tenants' IRR could be a strategy for allowing them in or not. For example, if the tenant's IRR is high and has little

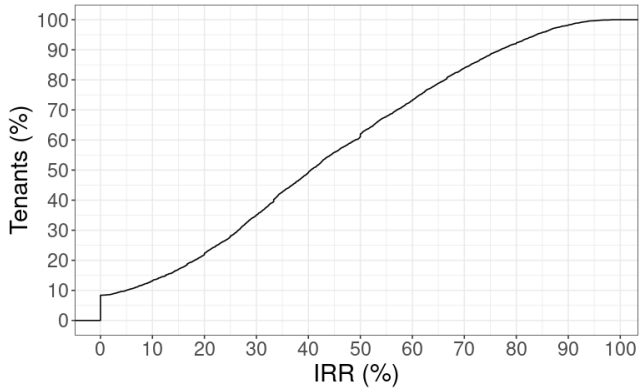

Figure 12: Cumulative distribution of IRR. Some tenants are better at capitalizing on caching advantages, while others may not demonstrate cache-friendly behavior, occupying memory and worsening performance.

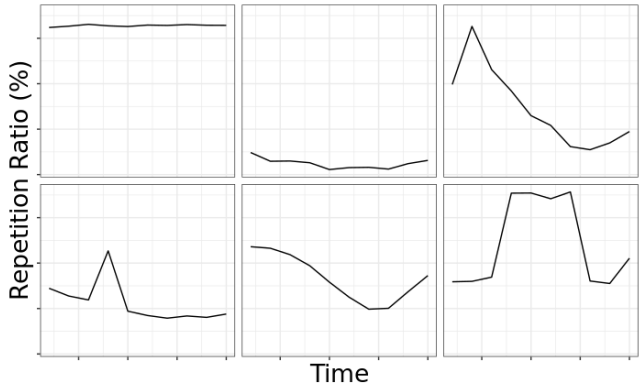

Figure 13: IRR over time for a selected sample of the biggest tenant by number of requests. Shows the many different patterns and variations by hour.

variation, it is a good candidate for cache entry. On the other hand, if the IRR is low and with little variation, it is a candidate for not entering the cache at any time.

In addition to the tenants with a slightly variable IRR, we also observed some that showed temporal oscillation. This dynamic behavior emphasizes the importance of evaluating the adequacy of the cache at specific time intervals. This approach makes it possible to discern tenants whose caching behavior may fluctuate over time, thus requiring adaptive cache management strategies for optimal performance. In this case, the system should ideally have a mechanism for calculating IRR periodically that plays a central role in fine-tuning cache capacity by selecting which tenant should or should not be in the cache at a given time.

4.5 Temporal Locality

This section will look at Inter-Reference Time. This helps us understand the tendency of the same item to be referenced frequently within short intervals. In general, we observe that the pattern of

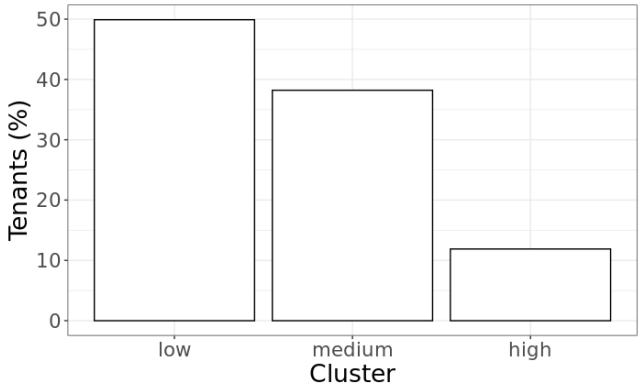

Figure 14: Percentage of tenants by a group based on IRR standard deviation, using the K-means algorithm. Around 50% of the tenants have a low standard deviation, while the other half have a more significant variation in the IRR.

temporal locality changes over time. This shows that tenants' and items' cache requests are variable over time. This further reinforces the need for dynamic management.

Many items have Mean Inter-Reference Time (MIRT) lower or equal to 150 minutes, specifically 57.20% of them. This indicates that the items are accessed with a relatively high frequency in a relatively short time. Table 3 also reinforces this by showing that the measures of central tendency (mean and median) are low. This is an indication of good cache usage by the items. On the other hand, 42.80% of the items have a Mean Inter-Reference Time longer than 150 minutes. This indicates that many items are idle during specific periods, taking up space in the cache.

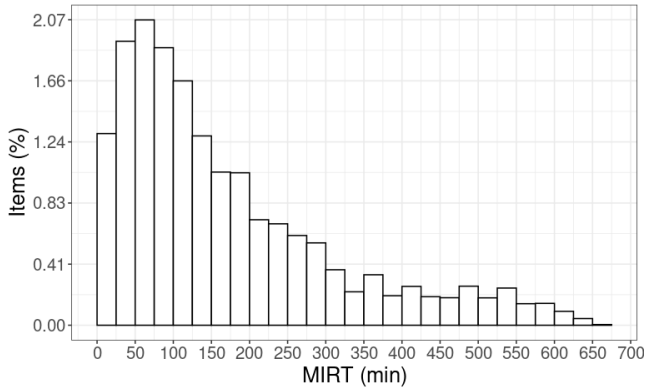

Figure 15: Percentage of items by Mean Inter-Reference Time (binwidth = 25). Most have MIRT lower or equal to 150 minutes. This points to the fact that the items are frequently accessed within a relatively short period.

Observing the MIRT of all items over ten consecutive windows of data can provide insights into the evolution of access patterns over time by observing how the frequency of access to items changes

Table 3: Statistical measures of MIRT for full workload.

Mean	Median	Max	Min
0.26	0.19	1	0

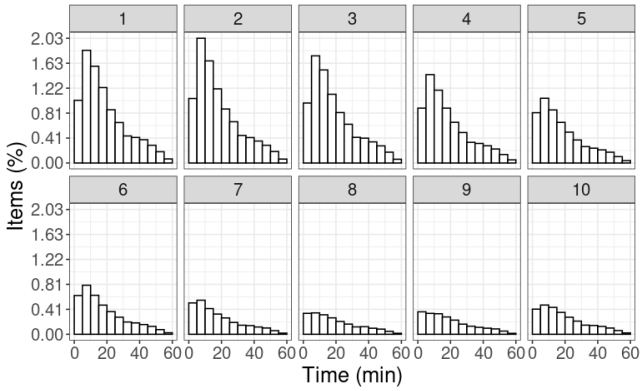

Figure 16: Percentage of items by MIRT for each 1-hour window to all items (binwidth = 5). Shows the dynamic trend in item access patterns.

over different periods. Figure 16 shows the dynamic trend in item access patterns.

During the first few hours, the number of access is high, and the average number of hits is concentrated mainly between 10 and 15 minutes. After that, the number of accesses decreases and becomes increasingly dispersed, although the bigger number of hits is still concentrated between 10 and 15 minutes.

5 PERFORMANCE IMPLICATIONS AND ISSUES

The observed load levels and access patterns notably influence the cache management and resource allocation in a multi-tenant environment. This section delves into the ramifications of the varied behaviors exhibited by tenants, highlighting possible strategies to optimize cache performance and mitigate resource over-provisioning.

5.1 Items with low frequency of access: A Case for Exclusion from Cache

Items with low repetition may not justify their inclusion in the cache. Allocating cache resources to items with sporadic access may lead to sub-optimal resource utilization. For example, for the cache system studied, 50% of the items stored correspond to only 20% of the requests. This means that they have a low frequency of access while they may occupy fifty percent of the resources. In this case, adopting admission control policies like Lazy Adaptive Cache Replacement [5] and CacheSack [9], that do not allow items with potentially low access to enter can be efficient strategies for cache management. These policies can maintain or improve system performance while demanding less allocation space.

5.2 Item Repetition Ratio as a Filter for Tenants

As Section 4 indicates, some tenants consistently exhibit a low Item Repetition Ratio. In such cases, for better cache utilization, items from these tenants may not be included in cache at any time. To control that, the IRR can work as a filter metric for tenant admission policy in the production environment.

We calculated the optimal hit ratio value for the entire workload, this value is approximately 59%. Based on that, we systematically performed an exclusion analysis of tenants with IRR values below specific thresholds to observe the optimal capacity that permits the cache reach the mentioned optimal hit ratio. This methodical approach produced convincing results, showing a notable increase in cache efficiency. When tenants with 20% or fewer IRR values were excluded, the optimal cache capacity was reduced to 90% of the original capacity, resulting in an optimal hit ratio of 61%. Similarly, imposing a stricter criterion of 30% IRR led to an optimal capacity of 78% of the original, culminating in a higher hit rate of 64%. Adopting a less conservative stance and excluding tenants with IRR values of 50% or less further improved the efficiency of the cache, producing an optimal capacity of 49% of the original capacity and a commendable hit rate of 71%.

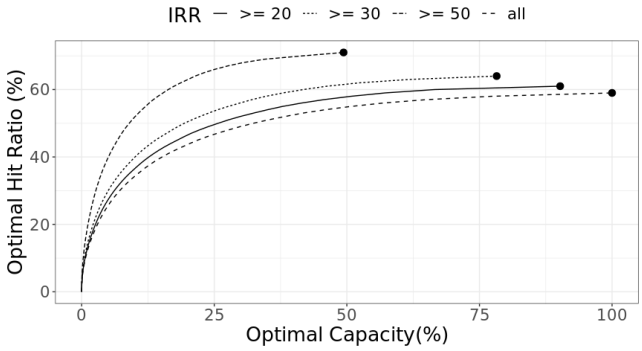

Figure 17: Optimal hit ratio by Optimal Capacity for each one of workloads. Each line is one type of workload (i.e. the line IRR >= 20 means that only tenants with IRR above or equal to 20 are on the workload). This suggests that some tenants with low IRR can be removed to reduce the capacity while maintaining the same level of performance.

These empirical findings imply that finding an IRR threshold can be an efficient way of controlling tenant admission to the cache system. Therefore, we can see capacity savings and a marked enhancement in hit ratios. This approach can not only optimize cache performance but also guarantee a heightened quality of service.

One potential strategy to manage cache usage by tenants involves analyzing Item Response Rate (IRR) and related metrics. This approach employs predefined threshold values to make decisions regarding whether to cache items from specific tenants. The primary objective is to optimize cache utilization by accommodating tenants with favorable IRR patterns while restricting cache storage for those with lower IRR. This approach can operate within defined time intervals, allowing tenants with temporarily low IRR to eventually regain cache access.

Another strategy is to identify tenants with undesirable characteristics and permanently exclude them from cache storage using URI identification and NGINX features. This approach aims to provide a directly and long-term control over cache access, ensuring that certain tenants do not impact cache performance negatively. These strategies collectively contribute to improving system performance and resource allocation.

5.3 Cache Resource Scaling

The dynamic nature of the Number of Requests, footprint, and Item Repetition Ratio for certain tenants highlights the need for dynamic capacity management of cache resources. This approach is akin to auto-scaling in virtual machines (VMs) or container orchestration platforms like Kubernetes. Just as auto-scaling adapts the number of VM instances or containers in response to real-time demand, cache resources must be flexible to accommodate evolving access patterns.

For example, during peak hours or promotional events, there may be a sudden increase in access to various items. This surge in demand requires a temporary boost in cache resources to maintain optimal performance. Conversely, maintaining the same cache capacity during off-peak periods is inefficient, as access frequency decreases. In such scenarios, dynamically scaling down the cache capacity can free up resources for more critical tasks. This dynamic resource allocation strategy not only optimizes cache performance but also enhances system efficiency and cost-effectiveness, aligning it with contemporary cloud-native computing paradigms.

6 CONCLUSION AND FUTURE WORK

This paper has provided an analysis of the workload characteristics in a multi-tenant cache environment using data from a prominent web cache service in a large-scale ecommerce platform. The findings have revealed a significant imbalance in request distribution among tenants, with approximately 80% of requests directed towards 10% of them. Moreover, we observed distinct access behavior patterns, ranging from steady loads to periodic spikes in request volumes.

Furthermore, the study highlighted that many requests concentrated on a few items, while others experienced sporadic access. Additionally, it was evident that some tenants demonstrated low cache-friendliness, resulting in a cache hit ratio decrease due to infrequent item retrievals.

The cache-friendliness is mainly related to tenants' request patterns and product inventory. Was realized important request patterns in tenants' workloads: Some tenants are more searched than others, and this implies a more significant number of requests to the system; can present seasonal request patterns, others sporadic patterns; can present peaks and/or valleys in request patterns, or can present stable patterns over time. In addition, there are tenants with an extensive number of products, which can mean that they need more storage space than others.

These load characteristics are important points for formulating effective cache management policies. What would be the impact of adopting a policy of partitioning resources between tenants to prevent large tenants from impacting smaller ones? What if we adopted a policy of admitting items? Raising the grain, what if we adopted a tenant admission policy? Also, what if we could set up a cache with dynamic capacity modification for adequate demand?

These management strategies will be our following objects of study. If they are indeed effective strategies, we will focus our efforts on implementing them in the production environment from an engineering point of view. In doing so, we aim to contribute to the overall efficiency and effectiveness of multi-tenant systems in real applications.

7 ACKNOWLEDGMENTS

First, I would like to thank my advisor, Thiago Emmanuel, for his guidance and unwavering support throughout this research. I am also deeply appreciative of my research group, particularly Ruan Alves and João Henrique, with whom I worked closely and who were already engaged in this cache universe. I extend my thanks to all the members of the Distributed Systems Laboratory, the laboratory that welcomed me and encourage my appreciation for science every day. I would like to thank Cleide for being the heart of the lab and for always being there for everyone.

I am immensely grateful to my family, who supports me throughout my academic journey. In particular, Flávia Lucena, Maria Júlia Lucena, Maria Eduarda Lucena, Josiete Lucena, and Eliete Lucena, your encouragement meant the world to me. I would also like to express my gratitude to my friends, especially the remarkable women scientists whom I am fortunate to call my friends. Sheila Paiva, Helen Cavalcanti, Andrielly Lucena, Leandra Oliveira, Narallynne Maciel, may we continue to champion the presence of women in science. Last but not least, I cannot forget to thank Mimi, the feline companion who kept me company during days of study and work.

This work has been funded by MCTIC/CNPq-FAPESQ/PB (EDITAL Nº 010/2021) and by VTEX BRASIL (EMBRAPII PCEE1911.0140).

REFERENCES

[1] Nginx documentation. https://nginx.org/en/docs/. Accessed: 2023-11-05.

[2] Arlitt, M. F., and Williamson, C. L. Internet web servers: workload characterization and performance implications. *IEEE/ACM Trans. Netw. 5*, 5 (1997), 631–645.

[3] Braun, H.-W., and Claffy, K. C. Web traffic characterization: an assessment of the impact of caching documents from ncsa's web server. *Computer Networks and ISDN systems 28*, 1-2 (1995), 37–51.

[4] Gu, R., Li, S., Dai, H., Wang, H., Luo, Y., Fan, B., Basat, R. B., Wang, K., Song, Z., Chen, S., Wang, B., Huang, Y., and Chen, G. Adaptive online cache capacity optimization via lightweight working set size estimation at scale. In *2023 USENIX Annual Technical Conference, USENIX ATC 2023, Boston, MA, USA, July 10-12, 2023* (2023), J. Lawall and D. Williams, Eds., USENIX Association, pp. 467–484.

[5] Huang, S., Wei, Q., Feng, D., Chen, J., and Chen, C. Improving flash-based disk cache with lazy adaptive replacement. *ACM Trans. Storage 12*, 2 (2016), 8:1–8:24.

[6] Megiddo, N., and Modha, D. S. ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA* (2003), J. Chase, Ed., USENIX.

[7] Vallamsetty, U., Kant, K., and Mohapatra, P. Characterization of e-commerce traffic. *Electronic Commerce Research 3*, 1 (2003), 167–192.

[8] Xiang, X., Ding, C., Luo, H., and Bao, B. HOTL: a higher order theory of locality. In *Architectural Support for Programming Languages and Operating Systems, ASPLOS 2013, Houston, TX, USA, March 16-20, 2013* (2013), V. Sarkar and R. Bodík, Eds., ACM, pp. 343–356.

[9] Yang, T., Pollen, S., Uysal, M., Merchant, A., Wolfmeister, H., and Khalid, J. Cachesack: Theory and experience of google's admission optimization for datacenter flash caches. *ACM Trans. Storage 19*, 2 (2023), 13:1–13:24.