Applied Mathematics and Nonlinear Sciences

<https://www.sciendo.com>

Multi-level cache management of quantitative trading platform

Zou Lida^{1,2,†}, Hassan A. Alterazi³, Mohammed Helmi Qeshta⁴

¹ School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China

² Key Laboratory of Machine Learning and Financial Data Mining in Universities of Shandong, Jinan 250014, China

³ Information Technology Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

⁴ Applied Science University, Kingdom of Bahrain

Submission Info

Communicated by Juan Luis García Guirao

Received March 2nd 2021

Accepted July 22nd 2021

Available online November 10th 2021

Abstract

With the rapid development of quantitative trading business in the field of investment, quantitative trading platform is becoming an important tool for numerous investing users to participate in quantitative trading. In using the platform, return time of backtesting historical data is a key factor that influences user experience. In the aspect of optimising data access time, cache management is a critical link. Research work on cache management has achieved many referential results. However, quantitative trading platform has its special demands. (1) Data access of users has overlapping characteristics for time-series data. (2) This platform uses a wide variety of caching devices with heterogeneous performance. To address the above problems, a cache management approach adapting quantitative trading platform is proposed. It not only merges the overlapping data in the cache to save space but also places data into multi-level caching devices driven by user experience. Our extensive experiments demonstrate that the proposed approach could improve user experience up to >50% compared with the benchmark algorithms.

Keywords: multi-level cache, cache placement, cached data merging, user experience function, quantitative trading platform

1 Introduction

Quantitative trading accounts for 70% of the total trading quota in a mature stock market. In China, the proportion of quantitative trading demonstrates rapid growth with the normalisation of the market. Quantitative trading platform is an important system of realising stock quantitative backtesting and deals. Due to the relative

[†]Corresponding author.

independence of investment deals and information technology, the researchers on stock trading strategy cannot independently develop and maintain a complete quantitative trading system. Therefore, quantitative trading platform is indispensable to validate and implement the proposed strategy.

Quantitative trading platform is divided into two types: private platform and public platform. Private platform is developed by fund firms or private-equity firms only for internal use, which has a relatively specialised function. Public platform is for public use, where users upload their own policies, perform backtesting and dealing. Public platform mainly gains profits by providing data and computation for users.

Massive quantitative trading platforms use two advantages to attract users: data quality and computing power. Especially computing power is critical for user experience and execution efficiency. Strategy backtesting is an important function of trading platform and it requires access to large amounts of historical time-series data, which is time-consuming. Therefore, the execution efficiency of strategy backtesting becomes an essential factor of user experience.

Cache is a critical device for improving data access efficiency. Latest time-series data are usually used when backtesting. It could largely reduce the time spent if the usually used data is placed into the cache. However, there are two issues in caching quantitative trading data. (1) Which data would be chosen and cached? Time-series data have their own characteristics: a user often accesses a piece of consecutive data, and the latest data would be frequently accessed and all the accessed data have including or overlapping relationships. Moreover, users on quantitative trading platform are divided into different types: VIP paying user, common user and experienced user. Different types of users have various expected backtesting performance and user experience functions with times. Therefore, the objective cached data depend on access characteristics of backtested data and user experience demands. (2) How are the cached data placed into multi-level caches? In stock quantitative backtesting, data are placed into storage devices such as memory and discs in the form of object intermediate result, which could improve access efficiency. There are many caching devices such as memory, network memory, solid-state disc and disc. They have different read-write access rates and capacities. The faster the access rate is, the smaller the capacity is. The devices fall into advanced cache and low-level cache and form a multi-level caching system. The proposed cache optimisation strategy is fit for all kinds of storage devices, and it no longer distinguishes between internal memory and external memory. The multi-level cache placement method needs researching for quantitative time-series data.

To address the above problems, we study multi-level cache management on quantitative trading platform in the paper. Cache-based data merging approach and cached data placement algorithm sensitive to user experience loss are proposed to improve user experience and access rates. The paper is organised as follows. Section 2 introduces the related works. The caching framework of quantitative trading platform is presented in Section 3. Section 4 gives cached data generation method based on data merging. In Section 5, a multi-level cache placement approach optimising user experience is designed. Sections 6 and 7 show the performance tests and conclusions, respectively.

2 Related works

With the increasing ratio of quantitative trading in stock trading, the design and optimisation of quantitative trading platform are becoming a concern. However, most works focus on trading strategy and trading data mining [1–3], while paying less attention to backtesting performance and user experience. Data access is an important part of quantitative platform backtesting, which can refer to optimising methods of user access experience. At present, the optimisation approaches of user access experience are experience value quantification [4], data placement [5–7], task scheduling [8–10] and cache management [11–13]. For cache management, the second exponential smoothing method is used to predict the probability distribution of user data accessed again, and then caching priority of each kind of user is determined [14]. Performance indicators of each kind of user are dynamically monitored and their caching spaces are adjusted according to real-time states [15]. A cache extension approach in a cloud computing environment is given to provide the isolated cache management model

for each kind of user, which could maintain a good access experience [16].

However, the data cache of quantitative trading has its own characteristics and demands, such as including an overlapping relationship of data access and multi-level cache, which the above works cannot be available for. In this paper, our work would focus on cache management of quantitative data.

3 User experience function and caching framework

User experience of quantitative platform is our optimisation objective of cache management. In the section, user access experience is quantified and a framework of cache management is introduced.

3.1 User experience function

User experience and return time of backtesting results show a negative correlation. For the convenience of denotation and computation, we assume that user experience is bad when experience value is large. Experience value is positively related to time. Since there are different types of users, experience function for each kind of user is distinct. For example, we can assume that experience value of users who pay increases fast with time and that of free users grows slowly or even stays unchanged. When making agreements with users, user experience functions are different. The main functions are shown in Figure 1. Assume that $f(t)$ is the user experience function, where t is the return time of backtesting results. In Figure 1(a), it is a linear function and formalised as $f(t) = a \cdot t$, where a is the slope. In Figure 1(b), it is a piecewise function and formalised as

$$\text{as } f(t) = \begin{cases} c_1 & 0 < t \leq t_1 \\ c_2 & t_1 < t \leq t_2 \\ c_3 & t_2 < t \end{cases}, \text{ where } c_1, c_2, \text{ and } c_3 \text{ are constants, and } t_1 \text{ and } t_2 \text{ are segmentation times. In Figure 1(c), it is a combination of linear and piecewise functions and formalised as } f(t) = \begin{cases} \text{at} & 0 < t \leq t_1 \\ at_1 & t_1 < t \leq t_2 \\ c_3 & t_2 < t \end{cases}.$$

In Figure 1(d, e) they are combinations and separately formalised as $f(t) = \begin{cases} c_1 & 0 < t \leq t_1 \\ c_2 & t_1 < t \leq t_2 \\ c_2 + a(t - t_2) & t_2 < t \end{cases}, f(t) = \begin{cases} c_1 & 0 < t \leq t_1 \\ \frac{(c_3 - c_1)}{(t_2 - t_1)}(t - t_1) & t_1 < t \leq t_2 \\ c_3 & t_2 < t \end{cases}$, where $\frac{(c_3 - c_1)}{(t_2 - t_1)}$ is the slope of Figure 1(e). In Figure 1(f), it is a non-linear function

$$\text{and formalised as } f(t) = a^{t-c} | a > 1, \text{ where parameter } a \text{ can adjust the growth rate of experience value and parameter } c \text{ controls the growth rate of the initial value. The experience value increases fast as time is large.}$$

Several common user experience functions are given above and the platform can use various functions as well. Different parameters are set to determine distinct experience loss rate according to user categories. Experience loss rate denotes the decline rate of user experience over time, which quantifies the effects of backtesting return time on experience value. The function provides a basis for multi-level cache management.

Different users have diverse demands for query time. Moreover, different weights are used in user experience functions in terms of user importance. The above way could unify the optimised objective. In addition, user experience is related to the profit of the service provider. The global objective of optimising user experience is the same as maximising the profits of the service provider.

Different users have diverse demands for query time. Moreover, different weights are used in user experience functions in terms of user importance. The above way could unify the optimised objective. In addition, user experience is related to the profit of the service provider. The global objective of optimising user experience is the same as maximising the profits of the service provider.

3.2 The framework of multi-level cache management

The framework of multi-level cache management on quantitative trading platform is demonstrated in Figure 2, which periodically manages cached data. The platform predicts the access volume of different data during the next period and adjusts the cached data in multi-level caching devices according to access volume and user

Fig. 1 User experience function.

experience function. When a user accesses data based on quantitative trading platform, the cached data are first obtained. If cached data cannot be fetched, it searches data from the database.

The cached data are stored in permanent storage media in the form of software function, i.e. they are stored into memory or disc by means of object or object serialisation. The caching characteristics of the software function are always returning the same results for the same input parameters, just like the decoration function 'lru_cache' in Python.

In theory, storage is finished by assigning storage space for cached data. For the convenience of adding, deleting and merging operations, the fixed amount of storage is set for each kind of caching device. Since the value and space of different caching devices are distinct, the fixed sizes for different devices are variously set. Generally, the fixed size of a device is proportional to the total storage space and inversely proportional to the times of additions and deletions on the device.

4 The generation of cached data

Before placing data into multi-level cache, the cached data and their value need to be determined. Access volume is an important factor of data value. In Section 4.1, we introduce the prediction method of access volume in future period. When users access time-series data of the same stock, different accesses would return part of overlapping data. It is an efficient way of saving caching space by merging the overlapping data into cache. Section 4.2 presents the merging approach of time-series data.

4.1 Access volume prediction in future period

In Figure 2, the module of access volume record records user access volumes of different data in each period. Based on historical data, the access volume of the next period can be predicted using statistics or machine learning algorithm.

When performing backtesting, time-series data are accessed generally by the function: $getData(t_b, t_e, s)$, where t_b is the start time, t_e is the end time and s is the target of stock data. After time-series data are fetched, filter operation is further executed. It is observed that the triple $\langle t_b, t_e, s \rangle$ could only determine a set of cached data. If $n_{b,e,s}$ denotes access volume of the data determined by a triple and $h_{b,e,s}$ is the amount of space taken up

Fig. 2 The framework of multi-level cache management.

by the data, a piece of data on the platform is formalised as $\langle t_b, t_e, s, n_{b,e,s}, h_{b,e,s} \rangle$.

Access volumes of different cached data next period can be predicted based on historic data. The common ways are historical statistics and machine learning. Historical statistics get the average value or weighted average value as a prediction value. Assume that q_t is the access volume to be predicted. We know $q_t = \sum_{i \in [1,n]} \alpha_i q_{t-i}$, $\sum_{i \in [1,n]} \alpha_i = 1$ stands, where q_{t-i} is the historical access volume of the i th period before current period. It is obtained through statistical data of the first n periods. Data in each period have a weight of α_i . Machine learning can use RNN, LSTM and GRU algorithms, and the prediction approaches are not our concerns.

4.2 Time-series data merging

Performing data merging for the overlapping part can save cached space. We assume that there are two pieces of data: $e_1 = \langle t_1, t_3, s, n_{1,3,s}, h_{1,3,s} \rangle$ and $e_2 = \langle t_2, t_4, s, n_{2,4,s}, h_{2,4,s} \rangle$, where $t_1 < t_2 < t_3 < t_4$ stands. The cached data determined by $\langle t_1, t_3, s \rangle$ and $\langle t_2, t_4, s \rangle$ have an overlapping part. As shown in Figure 3, the shadow area denotes the overlapping part. It is easily seen that data merging can save more space than that saved by separately caching. Without loss of generality, we use e_3 to denote the item after merging e_1 and e_2 , and formalised as $e_3 = \langle t_1, t_4, s, n_{1,4,s}, h_{1,4,s} \rangle$, $n_{1,4,s} = n_{1,3,s} + n_{2,4,s}$, $h_{1,4,s} = h_{1,3,s} + h_{2,4,s} - h_{\text{overlap}}$, where h_{overlap} is the size of overlapped data.

The method of determining data overlapping is as follows. (1) All the required data to be accessed are segmented and stored logically. (2) Assume that s_i is the identification of a piece of stock data. Logical hash index for s_i is constructed. (3) Logical hash index links to physical address of different levels of caching devices. (4) Based on hash indexes and their links, it is easy to determine whether there are overlapped data on the same level.

Since the determination of overlapped data is based on the hash index, its time cost is small. When the cached data are overlapped, the data on the same level of caching device are deleted directly.

Merging operation would change the value of cached data. Value is positively correlated with access value and negatively correlated with data size. Ignoring the other factors, the value of cached data can be computed as

Fig. 3 Time-series data merging.

$p_{b,e,s} = \frac{n_{b,e,s}}{h_{b,e,s}}$. After merging, the value of $p_{b,e,s}$ may increase or decline. Let us consider the following example. There are two data segments $n_{1,3,s} = 6$, $h_{1,3,s} = 6$, $p_{1,3,s} = 1$ and $n_{2,4,s} = 12$, $h_{2,4,s} = 6$, $p_{2,4,s} = 2$. When data merging is done and the space is $h_{1,4,s} = 10$, we get $p_{1,4,s} = (n_{1,3,s} + n_{2,4,s})/h_{1,4,s} = 1.8$, where its value is lower than $p_{2,4,s}$. If the space is $h_{1,4,s} = 6$, we get $p_{1,4,s} = (n_{1,3,s} + n_{2,4,s})/h_{1,4,s} = 3$, where its value is higher than $p_{1,3,s}$ and $p_{2,4,s}$.

$p_{b,e,s}$ can determine whether the cached data are placed into cache and which kind of cache they are placed into. For the case of $p_{1,3,s} > p_{1,4,s}$, i.e. the value $p_{1,4,s}$ after merging cached data $\langle t_1, t_4, s \rangle$ is smaller than the value $p_{1,3,s}$ of cached data $\langle t_1, t_3, s \rangle$, merging operation may lead to failing to be placed into cache some data that could have been cached. Therefore, our proposed merging algorithm conforms to the rule of not reducing value. The value after merging should not be smaller than that of any data before merging.

Based on the above considerations, we introduce our stock merging algorithm of cached data, which is called the value-first cached data merging algorithm. It first tries to merge data with high merging values, then places the merged data into cache and continually do merging until all the data cannot be merged again. Its concrete steps are shown in Algorithm 1, which just gives the merging method for a stock. Since the data are not overlapped between stocks, merging operations for other stocks also use Algorithm 1.

When a user accesses a set of time-series data and data are cached using the merging method, the data need to be restored. Our method is to scratch the required data in memory. Taking python as an example, cached data are first converted into DataFrame and then the selection is done. Since interception operation of data is easy and done in memory, the time lost accessing data from merged data and original data can be ignored.

5 Multi-level cache placement algorithm

Just as demonstrated in Figure 2, after the set of cached data C are prepared, they should be placed into multi-level cache to improve user access experience. Next, we discuss cache placement algorithm.

We assume that e_i denotes a kind of caching device, w_i denotes its capacity and $E = \{e_i, \dots\}$ denotes a set of the devices. The device with a high accessing rate is advanced cache and that with a low rate is low-level cache.

We next propose a multi-level cache placement algorithm oriented to user experience. It aims to optimise the user experience and evaluate the value of cached data based on user experience function, data size and accessing frequency. The data with high caching value are preferred to be placed into the advanced cache. The value of cached data c_j is computed as follows.

$$x_j = \frac{\sum f(t_{j,e_{i+1}}) - \sum f(t_{j,e_i})}{h_j} \quad (1)$$

where x_j is the estimated value of c_j , t_{j,e_i} is query time if putting c_j into e_i , $f(t_{j,e_i})$ is to compute user

Algorithm 1 Cached Data Merging Operation (CDMO)

Input: C, P . C is the set of cached data for a stock s and formalized as $C = \{c_i \mid i \in [0, n)\}$,

where i is integer. Assuming that p_i is the value of c_i , P is formalized as

$P = \{p_i \mid i \in [0, n)\}$, where i is integer.

Output: new value of C and P after merging.

```

1.  $O \leftarrow P$ 
2. WHILE  $O \neq \emptyset$  DO
3.    $p_k = \max\{p_i \mid p_i \in O\}$ ,  $O \leftarrow O - \{p_k\}$ 
4.    $Q \leftarrow O$ 
5.   WHILE  $Q \neq \emptyset$  DO
6.      $q_j = \max\{q_i \mid q_i \in Q\}$ ,  $Q \leftarrow Q - \{q_j\}$ 
7.     IF  $c_k$  and  $c_j$  overlap
8.       merging  $c_k$  and  $c_j$  and the new data is  $c_l$ 
9.       IF  $p_l \geq p_k$ 
10.         $C \leftarrow C - \{c_k, c_j\} + \{c_l\}$ 
11.         $P \leftarrow P - \{p_k, p_j\} + \{p_l\}$ 
12.      RETURN CDMO( $C, P$ ) // Iteratively call Algorithm 1
13.   END WHILE
14. END WHILE
15. RETURN  $C, P$ 

```

experience, $\sum f(t_{j,e_i})$ is the sum of all experience value if just putting c_j into e_i , h_j is the size of c_j and e_{i+1} denotes the caching device one level lower than e_i . Note that different users have distinct user experience functions.

The proposed algorithm is shown in Algorithm 2. It lets each level of caching device maximally improve the user experience. The data with a large improvement degree for user experience are placed into advanced cache, while those with small improvement are placed into low-level cache or not cached.

6 Performance evaluation

In this section, we validate the performance of the user experience of the proposed cache merging and placement algorithms. Our evaluation environment is set according to different combinations of devices and device

Algorithm 2 Multi-level Cache Placement Algorithm (MCP)

Input: C, E, W , where W is the set of device capacities, i.e., $W = \{w_i \mid e_i \in E\}$

1. $E \leftarrow \text{sort}(E)$ //sort devices in decreasing order of read-write rate.
2. FOREACH e_i in E
3. FOREACH c_j in C
4. Compute value x_j of c_j
5. $X \leftarrow X \cup \{x_j\}$ //put x_j into set of data value
6. END FOREACH
7. get the largest x_j in X , place c_j into e_i , until w_i runs out; update C , i.e.,
remove the cached c_j in C ; $X \leftarrow \emptyset$
8. END FOREACH

capacities. Its comparison indicators are average user experience value, average access rate and improvement degree of user experience.

6.1 The effect of cache capacity on user experience

Fig. 4 The effect of cache capacity on user experience.

In the experiments, we test how cached data merging operation (CDMO)-multi-level cache placement (MPC) performs in user experience for different cache capacities. CDMO-MPC means that Algorithm 1 is first used to merge cached data and then the data are placed using Algorithm 2. Two kinds of users are introduced: advanced

and common. Functions in Figure 1(a) and (e) are, respectively, their user experience functions. The indicator is the average value of user experience. Our proposed MPC algorithm is compared with two classical algorithms: Least Recently Used (LRU) and Least Frequently Used (LFU). Before making comparisons, the two algorithms are improved as follows. (1) First place data with high priority into the advanced cache. (2) The replaced data from the advanced cache are placed into the next lower level of cache. (3) The replaced data from the upper level of cache are placed into the next lower level, and the rest can be done in the same manner.

We use three caching media: memory, solid-state disc and disc and set their ratio of capacities as 1:10:100. In Figure 4, the horizontal ordinate represents the capacity of memory and that of the other caching devices that can be computed. We can see that MPC performs largely better than LRU and LFU in user experience, especially for small caching capacity.

6.2 The effect of caching device on user experience

Fig. 5 The effect of caching device on user experience. LRU, least recently used; LFU, least frequently used.

In this section, the effect of different cache combinations on user experience is validated. We separately set three combinations of caching devices: just memory, memory-SSD and memory-SSD-disc. As shown in Figure 5, MPC is >50% better than the other algorithms in the average value of user experience, no matter for which cache combinations. The main reason is that our proposed MPC algorithm aims to optimise the user experience. The user experience of MPC improves 61% than LFU in a just-memory environment. In the memory-SSD environment, the user experience of MPC improves 75%. In the memory-SSD-disc environment, it improves 81%. As shown in experiments, the MPC approach has a better optimising performance for the case of multiple caching devices.

6.3 The performance validation of merging algorithm

In this section, we test the CDMO algorithm. Four different combinations of algorithms are introduced. CDMO-MPC is to first use CDMO to merge data and then use MPC to place data. NoMerge-MPC is directly using MPC to place data without merging. CDMO-LRU is to first use CDMO and then execute LRU. NoMerge-LRU is to directly use LRU to place data without merging operation. The horizontal ordinate is memory capacity and the ratio of memory, SSD and disc is 1 10 100. From Figure 6, it is seen that both CDMO-MPC and CDMO-LRU perform better than their corresponding algorithms without merging operation, especially when caching capacity is small. In addition, NoMerge-MPC has a better user experience than CDMO-LRU, since MPC plays a fairly good role in optimising user experience.

Fig. 6 The performance of the merging algorithm. LRU, least recently used; LFU, least frequently used.

6.4 The performance comparison of different users

Fig. 7 The performance comparison of different users.

Here, we demonstrate the effects of user experience and access rate of MPC for different users. Advanced users and common users are set, and for user experience functions, refer to Section 6.1. Our comparison object is LRU. Figure 7 shows that common users have a better experience than advanced users for MPC, while LRU has little difference for different users. That is because MPC is driven by user experience. For accessing rate, MPC also shows a larger difference than LRU for two kinds of users, and MPC has a slower accessing rate for the common user than LRU. That is due to much more resources taken up by the advanced users.

7 Conclusions and future work

In the paper, we propose a cached data merging method and multi-level cache placement algorithm, which aim to optimise the user experience. Our demonstrated experiments reveal that they show good performance in improving user experience on quantitative trading platform, and multi-level cache placement algorithm mainly plays a critical role. It could optimise experience value for any kind of users, especially for advanced users.

Since there are massive users on quantitative trading platform and it provides backtesting and trading services based on a distributed environment, we will focus on cache management of quantitative trading platform in a distributed computing environment in future.

Acknowledgments

This work was supported by the Science and Technology Plan for Colleges and Universities in Shandong Province (KJ2018BAN046, KJ2018BAN058) and the 2020 annual project of the 13th Five-Year Plan for education science in Shandong Province (Grant No. 2020ZC312). The authors are grateful for the anonymous reviewers who made constructive comments.

References

- [1] Y. I Aji, 'An AHP-DEA-based vendor selection approach for an online trading platform,' *Int. J. Appl. Decis. Sci.*, vol. 6, no. 1, p. ied Decision Sciences, 2013.
- [2] L. Wei, H. Zhang, and T. An, 'B2C Trading Platform Security Strategy Based on the Optimal Stopping Theory,' *Commun. Comput. Inf. Sci.*, vol. 391, pp. 375–384, 2013.
- [3] C. Conlan, *Automated Trading with R: Quantitative Research and Platform Development*. 2016.
- [4] Lin C, Hu J, Kong X Z. Survey on Models and Evaluation of Quality of Experience[J]. Chinese Journal of Computers, 2012, 35(1):1-15.
- [5] 'Design and Implementation of A Replica Redundancy Management System Based on User Experience Value,' University of Electronic Science and technology, 2009.
- [6] Wu Y, 'User-experience-based availability analysis model and its application in P2P storage systems,' *China Academic Journal*, vol. 41, no. 9, pp. 1089–1100, 2011.
- [7] W. Yu, Y. Zhi, Q. Zhi, X. Zhen, and Y. F. Dai, 'User-experience-based availability analysis model and its application in P2P storage systems,' *Sci. China(Information Sci.)*, vol. 54, no. 8, pp. 1585–1595, 2011.
- [8] R. Chang, Y. C. Liu, J. M. Ho, Y. H. Chu, W. C. Chung, and C. J. Wu, 'Optimal scheduling of QoE-aware HTTP adaptive streaming,' in *Tencon IEEE Region 10 Conference*, 2016.
- [9] D. Zeng, G. Lin, G. Song, Z. Cheng, and Y. Shui, 'Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System,' *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, 2016.
- [10] D. Jie et al., 'User Experience Enhanced Task Scheduling and Processor Frequency Scaling for Energy-Sensitive Mobile Devices,' in *IEEE International Symposium on IEEE International Conference on High Performance Computing and Communications*, 2015.
- [11] S. Amini et al., 'Caching location-enhanced content to improve user privacy,' *Acm Sigmobile Mob. Comput. Commun. Rev.*, vol. 14, no. 3, pp. 19–21, 2010.
- [12] L. L. Fong et al., 'Toward a scale-out data-management middleware for low-latency enterprise computing,' *Ibm J. Res. Dev.*, vol. 57, no. 3/4, p. 6:1-6:14, 2013.
- [13] J. Huang, X. Liu, Z. Qi, J. Ma, and H. Gang, 'A browser-based framework for data cache in Web-delivered service composition,' in *IEEE International Conference on Service-oriented Computing and Applications*, 2010.
- [14] H. He, Z. Ma, H. Chen, W. Dan, H. Liu, and W. Shao, 'An SLA-Driven Cache Optimization Approach for Multi-tenant Application on PaaS,' in *IEEE Computer Software and Applications Conference*, 2014.
- [15] H. He, Z. Ma, H. Chen, and W. Shao, 'Towards an SLA-Driven cache adjustment approach for applications on PaaS,' in *Asia-pacific Symposium on Internetware*, 2013.
- [16] I. Stefanovici et al., 'Software-defined caching: managing caches in multi-tenant data centers,' in *Acm Symposium on Cloud Computing*, 2015.

This page is intentionally left blank