A Novel Adaptive Tuning Mechanism for Kafka-Based Ordering Service

Li Xu(✉), Xuan Ma, and Lizhen Xu

Department of Computer Science and Engineering,
Southeast University, Nanjing 211189, China
seu_lxu@163.com

Abstract. Thousands of resumes are sent to companies every year and it takes abundant time to authenticate the resumes. Blockchain (Hyperledger fabric) with its consensus algorithm, kafka-based ordering service, is a new solution to this issue, but it can not adapt to the dynamic workloads in real-time. This paper investigates an adaptive tuning mechanism based on feedback control theory to adjust the parameters connected to its consensus algorithm. In order to evaluate its efficiency, experiments have been done to compare the performance with the original kafka-based ordering service.

Keywords: Kafka-based ordering service · Hyperledger fabric · Adaptive mechanism

1 Introduction

Every year when it comes to graduate season, many well-known companies will receive thousands of resumes. It takes a lot of time and resources of the companies to identify the authenticity of the resumes. If the data of students' whole life cycle can be truly and untamable recorded such as awards record, certificates and grades, etc. a lot of time will be saved.

For that the blockchain [1, 2] is good at solving the problems of data security, sharing and reconciliation and so on, a lot of research has sprung up on the application industry of blockchain such as finance [3], medical care [4] and education [5, 6].

The sudden abundant data caused by students' collective activities requires fabric's consensus mechanism, kafka-based ordering service [7], should adapt to the current system workloads quickly. Therefore in this paper, a novel adaptive tuning mechanism (A-Kafka) contracting on adjustment of the parameters of the kafka-based ordering service based on feedback control theory [8] is proposed, with blockchain particularly Hyperledger fabric [9] utilized. In order to evaluate its efficiency, the proposed A-Kafka is studied on and analyzed in comparison with the original scheme of kafka (O-Kafka).

W. Ni et al. (Eds.): WISA 2019, LNCS 11817, pp. 119–125, 2019.
https://doi.org/10.1007/978-3-030-30952-7_14

2 Related Work

The consensus mechanism based on kafka is selected by Hyperledger fabric1.x, which has the characteristics of high throughput, low latency, extensibility, durability and reliability [10]. The kafka-based consensus mechanism contains kafka cluster and its associated zookeeper cluster, as well as ordering service nodes (OSN). Literature [11, 12] elaborated its consensus process:

Figure 1 shows the flow of kafka-based ordering service, when the transactions (e.g. tx1, tx2 … txn) created by clients, they will be broadcasted toward OSNs into kafka cluster. When the configurations of batchSize which is the maximum message count of the block and batchTimeout which is the maximum delay time from the last block generated are met, OSNs will package previously received transactions and generate a new block and after that, the block will be presented to committer peers which validate transactions in every block.

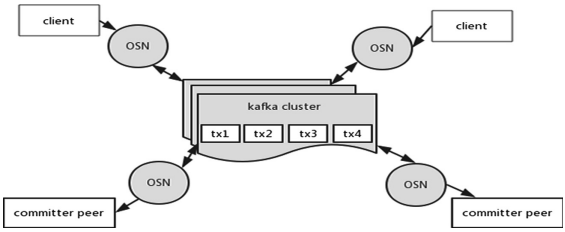

Fig. 1. The architecture of kafka-based ordering service

Currently, the configuration of O-Kafka about batchSize and batchTimeout could be set at startup [12], but the optimal policy configuration cannot be determined in real-time. Therefore, some researches related to the kafka-based consensus optimization have been carried out such as literature [13, 14].

3 Model

3.1 Model Representation

To describe the structure of A-Kafka, this paper models kafka-based ordering service into three stages: ordering, buffer and batching until configurations are met and committer validation.

A-Kafka continually evaluates the whole workload and the performance of consensus throughout a feedback control loop. It obtains information from kafka-based ordering service regarding the events such as pull transactions from the kafka cluster, buffer and cut block and process block (committer peers' validation) to adjust parameters of batchSize and batchTimeout.

3.2 Model Details

BatchTimeout

The batchTimeout should be large or small enough to meet the requirements of enough transactions batched or a busy state of the stage committer validation. MBV defined in this paper is the mean process time of buffer and batching and committer validation, and MOT defined is the mean time interval of OSNs pull transactions.

The Eq. (1) is the formula of MBV.

$$MBV_k = \alpha MBV_{k-1} + (1 - \alpha)BV_k \tag{1}$$

In the formula, α is the coefficient and $BV_k = T_{exe} - T_{start}$; T_{exe} is the instant time of complete validation of a block and T_{start} is the instant time when the OSN pulls a transaction from kafka cluster. MBV_{k-1} means the average value of the last T time intervals of MBV.

The Eq. (2) shows the formula of MOT.

$$MOT = \frac{t - t_0}{n} \tag{2}$$

In the formula, n represents the number of transactions that OSNs pulled in time interval $(t - t_0)$.

BatchSize

The adjustment method of batchSize is also should be large or small enough. The batchSize (BS) then could be calculated from the relationship of MBV and MOT and the Eq. (3) is presented below.

$$BS = \left\lceil \frac{MBV}{MOT} \right\rceil \tag{3}$$

3.3 Algorithms for A-Kafka

Algorithm 1 A-Kafka adjusting algorithm

Output: $BatchTimeout(BT)$, $BatchSize(BS)$

on event: OSN pulls a transaction **do**
  $T_{start} \leftarrow$ *the local clock timestamp*
  Number each transaction tx arrives OSN
on event: After committer peers validate transaction tx in block **do**
  search the transaction with the matched number
    $T_{exe} \leftarrow$ *the local clock timestamp*
    use T_{exe} and T_{start} compute MBV
on event: After OSNs pull tx from kafka-cluster **do**
  call Algorithm 3: BatchTimeout Adaptation
on event: After committer peers validate the last block **do**
  call Algorithm 4: BatchSize Adaptation

Algorithm 3 BatchTimeout Adaptaion

Input: MBV_k
Output: BT

compute MOT
if $MOT \geqslant MBV$ **then** $BS \leftarrow 0$
else $BT \leftarrow MBV_k$

Algorithm 4 BatchSize Adaptaion

Input: MBV, MOT
Output: BS

if fabric is startup **then** $BS \leftarrow default$
else $BS \leftarrow \lceil \frac{MBV}{MOT} \rceil$

4 Experiments

4.1 Objective

Latency (LT) and Throughput (TH) is widely used in performance evaluation. LT measures the mean time interval of a process from a transaction initially being created by clients to the transaction is confirmed by the blockchain. TH measures the numbers of transactions processed successfully number per second. What's more, PER $= \frac{TH}{LT*100}$ is utilized in this paper to evaluate the integral performance.

4.2 Experiments Design

The configuration of Hyperledger Fabric is fixed, the number of the nodes of zookeeper is fixed to 3, with that of the kafka nodes fixed to 4 and of the orderer nodes 3. And considering the influence factors of experimental performance are the size of transactions and the value of batchTimeout and batchSize. Therefore, the experiments are designed to those of fixed workloads and those of varying workloads.

4.3 Results and Analysis

Fixed-Workloads Experiments
The data collected from the fixed-workload experiments is shown from Tables 1 and 2. Obviously, the performance of A-Kafka is better than O-Kafka by observing variation of PER.

Table 1. Fixed-workloads results (5 clients, 8 KB)

Model	BS	BT(s)	LT(s)	TH(tx/s)	PER
O-Kafka	10	2	1.48	387.2	2.61
	10	30	1.53	395.0	2.58
	200	2	1.49	401.1	2.69
	200	30	0.73	536.9	7.39
A-Kafka			0.58	584.9	10.07

Table 2. Fixed-workloads results (50 clients, 8 KB)

Model	BS	BT(s)	LT(s)	TH(tx/s)	PER
O-Kafka	10	2	6.4	405.6	0.63
	10	30	5.5	413.0	0.75
	200	2	5.1	515.9	0.79
	200	30	3.7	838.6	1.01
A-Kafka			1.5	933.5	6.22

Varying-Workloads Experiments
Tables 3 and 4 present the results of the experiments of varying workloads with 5 and 50 clients engaged. We found that A-Kafka is more effective in experiments of varying workloads compared to those of fixed workloads by observing the change of PER.

To sum up, A-Kafka performs better than O-Kafka both in two types of experiments especially with varying workloads. To analyze, O-Kafka cannot cope with the changing workloads.

Table 3. Varying-workloads results (5 clients, 256B-8 KB)

Model	BS	BT(s)	LT(s)	TH(tx/s)	PER
O-Kafka	10	2	1.47	388.0	2.63
	10	30	1.39	389.9	2.80
	200	2	1.34	407.8	3.04
	200	30	0.54	542.6	10.01
A-Kafka			0.42	621.2	14.79

Table 4. Varying-workloads results (50 clients, 256B-8 KB)

Model	BS	BT(s)	LT(s)	TH(tx/s)	PER
O-Kafka	10	2	5.9	408.3	0.69
	10	30	5.0	416.6	0.83
	200	2	4.5	518.4	1.15
	200	30	3.2	841.9	2.63
A-Kafka			1.1	944.4	8.58

5 Conclusion

To address the difficulty in adapting to the dynamic workloads of the kafka-based ordering service in Hyperledger Fabric, this paper proposes an adaptive tuning mechanism A-Kafka. The results have shown that the effectiveness of this mechanism exceeds the O-Kafka, both in the experiments of fixed and varying workloads.

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2006)
2. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
3. Treleaven, P.: Blockchain technology in finance. Computer **50**(9), 14–17 (2017)
4. Wang, X., Hu, Q., Zhang, Y., Zhang, G., Juan, W., Xing, C.: A kind of decision model research based on big data and blockchain in eHealth. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 300–306. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_28
5. Schmidt, P.: Blockcerts-an open infrastructure for academic credentials on the blockchain. In: MLLearning (2016)
6. The Melbourne Newsroom. https://about.unimelb.edu.au/newsroom/news/2017/october/university-of-melbourne-to-issue-recipient-owned-blockchain-records. Accessed 23 June 2019
7. Kafka, A.: Kafka 0.9. 0 Documentation. http://kafka.apache.org/documentation.html#-introduction. Accessed 13 Apr 2016

8. Lu, C., Stankovic, J.A., Sang, H.S., Gang, T.: Feedback control real-time scheduling: framework, modeling, and algorithms. Real-Time Syst. **23**(1/2), 85–126 (2002)
9. Cachin, C.: Architecture of the hyperledger blockchain fabric. In: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, vol. 310 (2016)
10. Garg, N.: Apache Kafka. Packt Publishing Ltd, Birmingham (2013)
11. Christids, K.: A Kafka–based Ordering Service for Fabric. https://docs.google.com/documents/d/1vNMaM7XhOlu9tB_10dKnlrhy5d7b1u8lSY8akVjCO4. Accessed 2016
12. Hyperledger.org.: Hyperleger Fabric. https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html. Accessed 14 Apr 2019
13. Klaokliang, N., Teawtim, P., Aimtongkham, P., et al.: A novel IoT authorization architecture on hyperledger fabric with optimal consensus using genetic algorithm. In: 2018 Seventh ICT International Student Project Conference (ICT-ISPC), pp. 1–5. IEEE, Nakhonpathom, Thailand (2018)
14. Guo, Z., Ding, S.: Adaptive replica consistency policy for Kafka. In: MATEC Web of Conferences 2018, vol. 173, p. 01019. EDP Sciences (2018)