Comparative Analysis of Coordinated Checkpoint Protocol in Real-time Messaging System

Thandar Aung
NETsys Research Lab
University of Information Technology
Yangon, Myanmar
thandaraung@uit.edu.mm

Hnin Thiri Zaw
NETsys Research Lab
University of Information Technology
Yangon, Myanmar
h.thirizaw@uit.edu.mm

Aung Htein Maw
NETsys Research Lab
University of Information Technology
Yangon, Myanmar
ahmaw@uit.edu.mm

***Abstract*—In today's technological environments, the improvement of real-time messaging system also depends on the effectiveness of fault tolerance. The techniques of coordinated checkpoint protocol play a critical role in the development of real-time messaging systems. In this paper, we investigate the comparative analysis of coordinated checkpoint protocol by using Fixed Checkpoint Interval (FCI) method and Incremental Checkpoint Interval (ICI) method on real-time messaging system. In this study, the checkpoint interval protocol evaluates various batch sizes on Apache Kafka. The experimental results show the FCI method is more reliable than ICI method.**

Keywords—fixed checkpoint interval (FCI), incremental checkpoint interval (ICI), Apache Kafka, real-time

I. INTRODUCTION

Today, real-time processing is essential to improve the processing of IT industry work. Organizations need to enhance the performance of real-time processing more and more. In the distributed system environment, a popular technique for increasing dependability demand is fault tolerance. The effect of checkpoint interval impacts the development of fault tolerance. The reaction time of checkpoint interval is critical in real-time applications: traffic monitoring systems, healthcare, banking, or e-commerce.

Researchers evaluated the different techniques for fault tolerance in the real-time messaging system. Kumar et al.[1] compared the effective methods for various failures in the real-time distributed system. The author confirmed that the system is scalable, reliable and feasible by applying the fault tolerance method. Most of the systems consider the coordinated checkpoint protocol which is the effective method to define optimal checkpoint intervals with checkpoint cost, rollback overhead cost and total overhead cost. Checkpoints can be taken using either fixed checkpoint interval or incremental checkpoint interval [2]. In the case of fixed checkpoint interval, checkpoint interval size remains the same between any two successive checkpoints, but the incremental checkpoint interval methods take the different checkpoint interval sizes. It takes the second checkpoint interval size to be two times larger than the first one, the third checkpoint interval is three times more than the first one, and so on in each processing cycle. Daly, J, et al. [3] described by comparing the modified two models which provide accurate high-order approximations to the optimal checkpoint interval.

This paper emphasizes the high-performance checkpoint interval approach. The system examines and compares the two checkpoint interval methods (FCI and ICI). The system performs the reliability of a real-time messaging system by developing fault tolerance.

The paper is structured as follows: Section II explains coordinated checkpoint protocol on Apache Kafka pipeline architecture. Section III demonstrates the Performance Investigation on Apache Kafka. Finally, the system concludes in Section IV.

II. APACHE KAFKA PIPELINE ARCHITECTURE WITH COORDINATE CHECKPOINT PROTOCOL

Nowadays, data ingestion has become the critical process of real-time processing in the big data platform. Apache Kafka is a fault tolerant, reliable messaging system for data transferring in real-time. Fig.1. illustrates the system architecture of coordinated checkpoint protocol on Apache Kafka. The coordinated checkpoint protocol evaluates the processing flow of Apache Kafka.

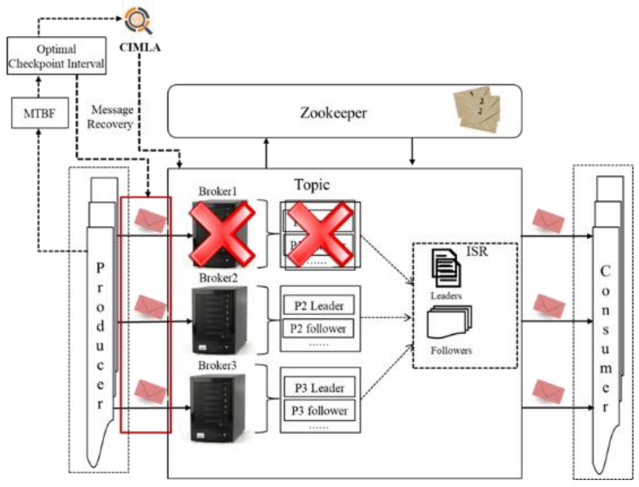


Fig.1. System Architecture of Coordinated Checkpoint Protocol on Apache Kafka

The system calculates the time between failure based on total uptime (uptime and downtime) and number of breakdowns that depend on the publishing batch size (file size). In Equation (1), the Mean Time Between Failure (MTBF) is the predicted time interval between system failures during real-time operation.

$$MTBF = \frac{Total\ uptime}{Number\ of\ breakdown} \quad (1)$$

The optimal checkpoint interval ($\mathrm{T_{opt}}$) is calculated using MTBF and the save time ($\mathrm{T_{sd}}$) of the Kafka configuration in Equation (2).

$$\mathrm{T_{opt}} = \sqrt{2 \times \mathrm{T_{sd}} \times \mathrm{MTBF}} \quad (2)$$

A. Fixed Checkpoint Interval (FCI) Method

Fig.2. illustrates the procedures for the FCI method as well as the number of checkpoints in the ith cycle. The system determines the ***starting time of the ith checkpoint*** ($\mathrm{C_i}$) based on $\mathrm{T_{opt}}$ and $\mathrm{T_{sd}}$ in Equation (3) to represent the total overhead cost of Apache Kafka.

$$\mathrm{C_i} = \mathrm{iT_{opt}} + (\mathrm{i}-1)\mathrm{T_{sd}} \quad (3)$$

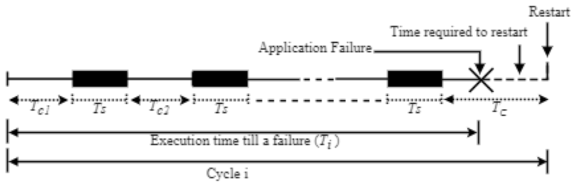


Fig.2. Procedures for FCI Method

In FCI method, the length of each checkpoint interval is fixed i.e., $\mathbf{T_{Ci} = T_C}$. The system assumes the ***save time*** (T_{sd}) of 5 seconds from the default save time from Apache Kafka and accepts ***failure time*** (T_{if}) from dynamic server failure time in the running process for overall calculation of the system. During failure recovery, the system calculates the total overhead cost due to checkpointing and restarting.

T_{if}, T_{opt} and T_{sd} are used in Equation (4) to compute the ***number of checkpoints taken in the ith cycle*** (N_i).

$$N_i = \left\lfloor T_{if} / (T_{opt} + T_{sd}) \right\rfloor \tag{4}$$

Equation (5) calculates ***checkpointing cost in the ith cycle*** ($\mathbf{CC_i}$) by using $\mathbf{N_i}$ and $\mathbf{T_{sd}}$.

$$CC_i = N_i T_{sd} \tag{5}$$

In Equation (6), $\mathbf{N_i}$ and ***optimal checkpoint interval and save time*** ($\mathbf{T_{opt} + T_{sd}}$) in Kafka. ***Rollback recovery*** ($\mathbf{Rb_i}$) has measured the difference between current server failure, the number of checkpoints to be taken in ***ith*** cycle and each checkpoint interval cost. The system needs to know the ***restart time*** ($\mathbf{T_{rt}}$) after the failure case, Kafka defines the restart time as 5 seconds by default.

$$Rb_i = (T_{if} - N_i(T_{opt} + T_{sd})) \tag{6}$$

In Equation (7) , the ***overall total overhead cost in the ith Kafka processing cycle*** $E(T_{cost})$ is determined by rollback, checkpoint, and restart costs.

$$E(T_{cost}) = CC_i + (T_{if} - N_i(T_{opt} + T_{sd})) + T_{rt} \tag{7}$$

B. Incremental Checkpoint Interval (ICI) Method

Fig.3. demonstrates the procedures of the ICI method and the number of checkpoints in the ith cycle. In each cycle the first checkpoint is initiated after T_{opt1}, the second checkpoint is initiated after ($T_{opt1} + T_{opt2} + T_{sd}$) and the third checkpoint is initiated after ($T_{opt1} + T_{opt2} + T_{opt3} + 2T_{sd}$) and so on. The checkpoint interval size of ICI method can vary from one checkpoint to another checkpoint. In each cycle, the increment of checkpoint interval size increases based on the initial checkpoint interval time. Thus, the rollback cost of ICI method can increase more than the rollback cost of FCI method.

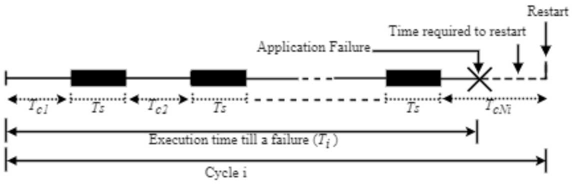


Fig.3. Procedures for ICI method

The system considers ICI for processing i.e., $\mathbf{T_{Ci} = iT_C}$ in followings and then calculates in Equation 8.

$$\left.\begin{array}{l} \text{If } T_1 < (T_{opt1} + T_{sd}), \quad N_1 = 0 \\ \text{If } \left(\sum_{k=1}^{i} K T_{opt} + nT_{sd}\right) < T_i \text{ and} \\ \text{If } Ti < \left(\sum_{k=1}^{i} K T_{opt} + (n+1)T_{sd}\right) \end{array}\right\} \begin{array}{l} N_i = n \text{ where} \\ n = 1,2,3,\ldots \\ i = 1,2,3, .. \end{array}$$

$$C_i = \sum_{k=1}^{i} T_{optK} + (i-1)T_{sd} \tag{8}$$

In Equation (9), the size of first checkpoint interval is T_{opt} that performs more than its previous checkpoint interval in each cycle.

$$N_i = \left\lfloor T_{if} / (T_{opt} + T_{sd}) \right\rfloor \tag{9}$$

Equation (10) calculates based on the Equation (9) and T_{sd} in kafka.

$$CC_i = N_i T_{sd} \tag{10}$$

Rb_i has measured the difference of current server failures and the number of checkpoints to be taken in ith cycle and each checkpoint interval cost. Rb_i can calculate by using parameters N_i, T_{if}, T_{opt} and T_{sd} in Equation (11). N_i is the number of checkpoints to be taken in ith cycle and ***the time of optimal checkpoint interval*** (KT_{opt}).

$$Rb_i = (T_{if} - (N_i T_{sd} + \sum_{k=1}^{i} K T_{opt})) \tag{11}$$

Rollback, checkpoint, and restart costs are used to calculate the total overhead cost in the ith Kafka processing cycle $E(T_{cost})$.

$$E(T_{cost}) = CC_i + (T_{if} - (N_i \times T_{sd} + \sum_{k=1}^{i} K T_{opt})) + T_{rt} \tag{12}$$

The system's reliability can be measured by using Equation (13) with the parameters (R_1, R_2, R_3,.., Rn). The system decides the whole system's reliability by evaluating the parallel system reliability.

$$R_s = 1 - (1 - R_1)(1 - R_2) \ldots (1 - R_n) \tag{13}$$

III. Experimental Studies and Performance Investigation

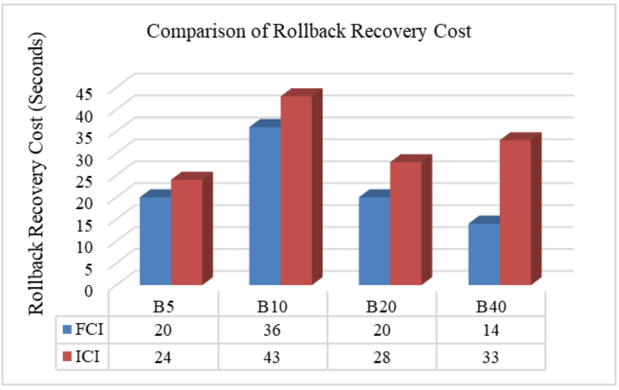
The performance of the coordinated checkpoint protocol has been evaluated on the Kafka pipeline architecture. The comparative analysis of the Fixed Checkpoint Interval (FCI) method and Incremental Checkpoint Interval (ICI) method are performed. The system tests the two experiments on the four types of batch size: B5, B10, B20 and B40 on data size (file sizes)184 MB from real-time messages in Kaggle Web site.

A. Experiment 1: Comparison of Rollback Recovery Cost on Various Batch Sizes

The system demonstrates the comparison of rollback recovery cost in Fig.4. The testing focuses on how the effectiveness of the FCI method and ICI method.

The results show the different rollback recovery costs among batch sizes. Reducing the rollback recovery cost is the main point for calculating the total overhead cost. The FCI method reduces the time complexity on different batch sizes than the ICI method.

Hence, the FCI method can implement any batch sizes in real-time messaging system.

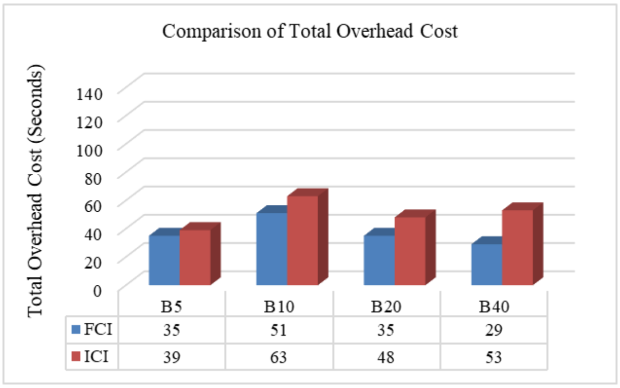


	B5	B10	B20	B40
🟦 FCI	20	36	20	14
🟥 ICI	24	43	28	33

Fig.4. Comparison of Rollback Recovery Cost in FCI and ICI Method

B.Experiment 2: Comparison of Total Overhead Cost on Various Batch Sizes

The system evaluates the comparison of total overhead cost in Fig.5. The total overhead cost is calculated based on the server failure time that varies the number of batch sizes (file sizes).



	B5	B10	B20	B40
🟦 FCI	35	51	35	29
🟥 ICI	39	63	48	53

Fig.5. Comparison of Total Overhead Cost in FCI and ICI Method

The results show FCI method outperforms for measuring the total overhead cost of the system. Hence, the FCI method reduces the total overhead cost compared to the ICI method.

Hence, the system recovers total overhead costs in more efficient time for different batch sizes.

C. Reliability Measure on Experimental Results

Table I summarizes the comparative analysis of coordinated checkpoint protocol based on the experiments.

TABLE I. RELIABILITY PERFORMANCE FOR EXPERIMENTS

Number of Batch size (184 MB)	**Roll Back Recovery Cost**	**Total Overhead Cost**
B5	17%	11%
B10	17%	20%
B20	29%	28%
B40	58%	46%

Table I shows the time complexity for the rollback recovery cost of FCI method absolutly reduces than the ICI method on different batch sizes (file sizes). Then, the total overhead cost of the FCI method is better than ICI method to measure the reliability of Apache Kafka.

The system analyzes the reliability improvement on the experiments by comparing FCI and ICI method. Hence, *the system reliability raises 79% in rollback recovery cost and 72% in total overhead cost by applying the FCI method.*

IV. CONCLUSION

Fault tolerance is the importance in the reliability of Apache Kafka. The development of Kafka's fault tolerance relies on a reliable checkpoint interval method. The system compares two checkpoint interval methods on different batch sizes. The analysis indicates the advantages of the Fixed Checkpoint Interval (FCI) method which reduces the recovery cost than the Incremental Checkpoint Interval (ICI) method. Hence, the FCI method is more appropriate for measuring the reliability of the real-time messaging system. The paper confirmed that the FCI method improves about 80% for the real-time messaging system.

REFERENCES

[1] Kumar, A., Yadav, R.S. and Ranvijay, A.J., 2011. “Fault tolerance in real time distributed system”. International Journal on Computer Science and Engineering, 3(2), pp.933-939.

[2] Silva, L.M. and Silva, J.G., 1999, April. “The performance of coordinated and independent checkpointing”. In Proceedings 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing. IPPS/SPDP 1999 (pp. 280-284). IEEE.

[3] Daly, J., 2003, June. “A model for predicting the optimum checkpoint interval for restart dumps”. In *International Conference on Computational Science* (pp. 3-12). Springer, Berlin, Heidelberg.

[4] Shastry, M.P. and Venkatesh, K., 2010. “Selection of a Checkpoint Interval in Coordinated Checkpointing Protocol for Fault Tolerant Open MPI”. International Journal on Computer Science and Engineering, 2(6)., ISSN: 0975-3397, Engg Journals Publications, India.

[5] Aung, T., Min, H.Y. and Maw, A.H., "CIMLA: Checkpoint Interval Message Logging Algorithm in Kafka Pipeline Architecture." In *2020 International Conference on Advanced Information Technologies (ICAIT)*, Yangon, Myanmar, pp. 30-35, November, 2020, IEEE Digital Library, DOI: 10.1109/ICAIT51105.2020.9261812.