An asynchronous consistency algorithm in smart manufacturing cloud data centers

Yang Lu\textsuperscript{1a}, Yan Zheng\textsuperscript{2b}, Weipeng Jing\textsuperscript{1c}, Changlin Xu\textsuperscript{1d}, Yan Li\textsuperscript{3e}

\textsuperscript{1}Department of Information and Computer Engineering, Northeast Forestry University, Harbin 150006, China
\textsuperscript{2}Department of Computer and Information Engineering, Harbin Normal University, Harbin 150500, China
\textsuperscript{3}Department of Platform Research, Heilongjiang NUT Technology Co., LTD, Harbin 150040, China
\textsuperscript{a}luyang@nefu.edu.cn, \textsuperscript{b}hitzhengyan@126.com, \textsuperscript{c}nefujwp@163.com, \textsuperscript{d}changlinx@163.com, \textsuperscript{e}liberty.li@jufengtools.com.cn

Abstract—In the smart manufacturing factory, the fresh real-time data and multiple replications are important for numerically-controlled machine tools. The production data are stored in distributed cloud data centers (CDCs), so the communication latencies produced by data synchronization may impair the machining performance. In this paper, an asynchronous consistency algorithm (ACA) which is inspired by the nearest infection is proposed to reduce the communication latencies between nodes. The experimental results demonstrate that the proposed algorithm can support lower client-side latencies and increase data stability, thereby improving the production efficiency.

1. Introduction

In a smart factory, the IoT (Internet of things) is applied to capture the data generated by the production facility, so big data technology is used to store and analyse data in real-time [1]. For this data-driven machine manufacturing, system control and management are supported by the data from the production workshop [2]. Cloud data centers (CDCs) provide the data storage and computing platform to smart factories. However, they will also face some new challenges, where the data management of latency-sensitive applications is one of them, such as data reading in numerical control, machining tools test, and interactive drawing editing. Different from other application scenarios, data in machine manufacturing are expensive. Thus more replicas are necessary and they should be placed in geo-distributed data centers (DCs) to collect the global data for predicted models. To avoid clients’ latencies, weak consistency scheme is the best solution. Some existing works focused on the optimization of the replicas synchronization and consistency scheme. In [3], a machine learning-based predictive framework is present, which can select client-centric consistency methods under the limit of latency and staleness thresholds. In [4], the notion of almost strong consistency as an option for the consistency/latency tradeoff is discussed, and the deterministically bounded staleness of data versions is demonstrated. In [5], the authors point out that weaker consistency models could be maintained when scalable services are deployed in partitionable systems. In [6], an adaptive scheme for consistency level at run-time according to the application requirements is proposed to achieve high data availability. In [7], a cross-area node selection algorithm is proposed to minimize the...
communication latency, and the tradeoff between reliability and synchronization time is discussed. In [8], the eventually consistent services are discussed when data-centric consistency models are not implicitly convergent. In [9], a replicas allocation algorithm is raised to preserve the server consistency in data-intensive CDCs, but the performances of clients are not considered.

In this paper, when the input datas come from several local nodes, we study to reduce the client-side latencies in CDCs for smart manufacturing. The major contributions of this paper are summarized as follows. The proposed ACA is not to completely control the number of replicas, so more replicas will be supported in some specific works. Based on ACA, the purpose is to guarantee the manufacturing under the limitation of the latency threshold. With asynchronous, we present an infectious consistency scheme, where the storage node selecting method is considered to reduce the latencies and guarantee the data safety.

2. Materials and Methods
Because the manufacturing factory may generate a massive amount of datas, so the storage system is a large-scale CDC with $N_{\text{node}}$ nodes, $N_{\text{rack}}$ racks, and $N_{\text{dc}}$ data centers. The set of nodes, racks, and DCs is described as $S = \{S_1, S_2, \ldots, S_{N_{\text{node}}}\}$, $R = \{R_1, R_2, \ldots, R_{N_{\text{rack}}}\}$, and $C = \{C_1, C_2, \ldots, C_{N_{\text{dc}}}\}$ respectively. The input data items are stored in the local nodes, and the replicas are sent to the remote ones. The data center framework is shown in Fig.1, in which the modules are deployed in the public and private cloud. The private cloud is the local DC in an enterprise, and the public cloud means the remote nodes. The layered architecture can increase the data collecting and processing efficiency.

![Fig.1 The large-scale data center in smart manufacturing factories](image)

We assume that the default replicas number is set as $M$, and the weak consistency factor is denoted as $\alpha$. When a data input request comes to the system in time $t_{iS}$, the server-side latency is defined as

$$L = \sum_{i=1}^{dM} t_{iS}$$  \hspace{1cm} (1)

We present the whole CDC as an undirected graph $G = (V, E)$, where $V = \{V_1, V_2, \ldots, V_N\}$ is the set of nodes. The degree of dispersion of the replicas is defined as $d$, and $A$ is the adjacent matrix of vertex set $V$. The elements in $A$ are the transmission distance of the data replicas. Similar to Hadoop, the distances between nodes are mapped to different levels: two programs in the same node, two nodes on the same rack, two nodes on different racks in the same DC, and nodes in different DCs, where the
corresponding distances are denoted as set \{0, 2a, 4a, 6a\}. We define the data storage matrix \(D\), in which each element \(d_{ij}\) in \(D\) is the distance between nodes. The degree of dispersion is described as

\[
d = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij})^2}
\]  

(2)

The optimization objectives of replicas storage are the minimum latency, consistency, and reliability. The consistency means the clients can get the fresh data when the system is available and the reliability performance is evaluated by \(d\). During the transmission of data package, round trip time (RTT) means the total transmission time for data sending and reception, which is constrained by network distances. When there are \(S\) equipments, the total transmission delay \(p\) [10] is

\[
p = ps \frac{1}{C_i} + \sum_{i=1}^{S} R_i D_i + \sum_{i=1}^{S} R_i \gamma_i + \sum_{i=1}^{S} q_i
\]  

(3)

In equation (3), \(ps\) is the packet size, and \(C_i\) is the channel capacity. \(R_i\) is transmission speed and \(\gamma_i\) is the difference between link and physical distance. The random dynamic noise is \(q_i\), and \(\frac{1}{C_i} + R_i \gamma_i\) is a fixed value. Based on the topology of backbone optical fibers, the network path distance between remote routers is not deviated from the actual physical distance. The relation between delay and distance is linear. Therefore, the latencies generated by replicas in distributed DCs are formulated as

\[
\text{minimize } \sum_{i=1}^{N} d_{ij}
\]

s. t. i) \(d_{ij} > 0\)

ii) \(d \in \{0, 2a, 4a, 6a\}\)

iii) \(\sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} (d_{ij})^2} \geq \sigma\)

(4)

According to equation (3), the latencies can be replaced by the normalization factor \(a\), which is denoted as

\[
a = \sqrt{\sum_{i=1}^{N} (A[i][j])^2}
\]  

(5)

Equation (4) illustrates the best replicas storage scheme with the minimum dispersion degree. When the data item is written into the local DC, the intuitional storage scheme is to choose the adjacent nodes. Meanwhile, data safety should be guaranteed by dispersive replicas. But the optimization problem is NP-hard. As a result, we present a heuristic asynchronous consistency algorithm, whose advantage is reducing the latency by synchronous process, and getting the system consistency by the asynchronous storage in remote nodes. According to the weak consistency requirement, the node selection sequence is from data center to nodes. The proposed replicas storage scheme is demonstrated in Alg.1 and Alg.2.

Based on equation (4), the replica should be stored into the nearest node, and then it will infect other ones. We focus on the infection in the adjacent nodes and dispersion in the remote DCs, which is proceeded asynchronously. In general, the local DC may first obtain the consistency requirement. As a result, the proposed ACA includes two write procedures. In Algorithm 1, step 1 acts as a proxy node to receive the data write requirement. Corresponding to the minimum latency target, except the weak consistency setting, the remaining replicas are sent to nodes in other DCs and the receipt is not
necessary (2-3). In steps (4-9), the nodes in remote DCs are selected to store the replicas in terms of Alg.2.

Algorithm 1 Asynchronous consistency algorithm

Input: The information of nodes in the whole data center, including the set of nodes $S$, racks $R$, and DCs $C$.

Output: The replicas’ write information.

1: if a client start a write request. then
2:  Calculate the weak consistency guarantee replicas number $\alpha M$ and one local node is selected as the initial storage node.
3:  Send the remaining replicas to the remote nodes.
4: while The number of written replicas is less than the set value. do
5:    Select nodes randomly in set $S_{s}$ by Alg.2.
6:    if The replicas are written to the nodes completely. then
7:      Notify the master node that the data update process is finished.
8:    else
9:      Perform the replicas update again.
10: return result

Algorithm 2 Infectious node selection algorithm

Input: The information of nodes in the whole data center, including the set of nodes $S$, racks $R$, and DCs $C$.

Output: The candidate node set $S_{e}$.

1: Calculate the eligible DCs according to the degree of dispersity and store them in set $S_{dc}$.
2: for all the DCs in $S_{dc}$ do
3:    for all the DCs in $S_{rack}$ do
4:      Get the adjacent nodes.
5:    return nodes set $S_{e}$.

In step (1), the DC selection is restricted by $d$, and the distances are considered in DCs. Then the nearest node is obtained by the infectious scheme in each rack of the decided DC in parallel (2-4). In step (5), the candidate node is returned for replicas writing in set $S_{e}$.

3. Test Results and Discussions

In our experiments, results display the advantage of performance in the degree of dispersity and latencies of the proposed ACA. The OS is centos 7, and the host IP is 192.168.100.0/24. The server number is 16, which is virtualized by docker containers. After testing, the communication time loss of containers is about 20%.

$N$ means the node number. The simulation is implemented with $N = 10$ and $N = 100$ in a cloud computing environment, which provides production data for manufacturing. Data write testing of the dispersity degree is shown in Fig.2. The latency threshold is set as 110 ms, so the dispersity is satisfied to the greatest extension. In the traditional database such as Cassandra or HDFS, the adjacent nodes are selected first. Because the replicas order of the proposed ACA starts with the DC, the dispersity performance is better. Meanwhile, when the number of nodes increases to 100, the fixed replications cannot promote dispersity.

The measuring of the average latency is shown in Fig.3. The replications of the manufacturing in factories are precious, so more storage content is allowed. Compared with the traditional scheme, the proposed ACA reduces the latency down to 39.1%. Besides, as the replications number increases, the
latency increases slightly. These advances of the proposed ACA, especially for a mass replications applications in manufacturing factories, are more appropriate.

4. Conclusion
In this paper, we focus on the data writing algorithm in cloud data centers of the smart manufacturing scenarios. The asynchronous write process is present to sustain the multiple replications. Based on the results and discussions presented above, the conclusions are obtained as below:

1) It is shown that the client-side latencies are minimized by the degree of dispersity limitation. The consistency requirement is obtained first in local DC, and then the replications are sent to other DCs like an infection.

2) Simulation results show that the proposed ACA obtains better performance, and more replications cannot extend the system latencies. In future work, the network bandwidth generated by the multiple replications should be considered.

Acknowledgments
This work has been supported by “the Fundamental Research Funds for the Central Universities”, No. 2572019BH01.
References

[1] C. Kim and S. Son, "A Study on Big Data Cluster in Smart Factory using Raspberry-Pi," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5360-5362.

[2] A. Sîrbu and O. Babaoglu, "Towards Data-Driven Autonomies in Data Centers," 2015 International Conference on Cloud and Autonomic Computing, 2015, pp. 45-56.

[3] S. Sidhanta, W. Golab, S. Mukhopadhyay and S. Basu, "Adaptable SLA-Aware Consistency Tuning for Quorum-Replicated Datastores," in IEEE Transactions on Big Data, vol. 3, no. 3, pp. 248-261.

[4] H. Wei, Y. Huang and J. Lu, "Probabilistically-Atomic 2-Atomicity: Enabling Almost Strong Consistency in Distributed Storage Systems," in IEEE Transactions on Computers, vol. 66, no. 3, pp. 502-514, 1 March 2017.

[5] F. D. Muñoz-Escoí, R. de Juan-Marin, J. García-Escrivá, J. R. González de Mendivil and J. M. Bernabéu-Aubán, "CAP Theorem: Revision of Its Related Consistency Models," in The Computer Journal, vol. 62, no. 6, pp. 943-960.

[6] B. Gueye, A. Ziviani, M. Crovella and S. Fdida, "Constraint-Based Geolocation of Internet Hosts," in IEEE/ACM Transactions on Networking, vol. 14, no. 6, pp. 1219-1232.

[7] Y. Lu, C. Xu and Y. Zheng, "Cross-area Node Selection Algorithm for Data Replica Storing in Cloud Data Centers," 2020 International Conference on Computer Engineering and Intelligent Control (ICCEIC), 2020, pp. 190-194.

[8] L. P. Miret and F. D. Muñoz Escoí. (2016) Replica Divergence in Data-Centric Consistency Models. 27th International Workshop on Database and Expert Systems Applications (DEXA), Porto, 2016, pp. 109-112.

[9] S. Basu and P. K. Pattnaik. (2017) A consistency preservation based approach for data-intensive cloud computing environment. The 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-5.

[10] Chengbo Jiao, Hui Zheng, Yu Huang. Journal of university of electronic science and technology of China, 2012, 41(05): 781-786.