Providing Persistence for Sensor Data Streams with Temporal Consistency Conscious WAL

Hideyuki KAWASHIMA
Science for Open and Environmental Systems
Keio University, Japan
email: kawasima@ayu.ics.keio.ac.jp

Motomichi TOYAMA
Information and Computer Science
Keio University, Japan
email: toyama@db.ics.keio.ac.jp

Michita IMAI
Information and Computer Science
Keio University, Japan
email: michita@ayu.ics.keio.ac.jp

Yuichiro ANZAI
Information and Computer Science
Keio University, Japan
email: anzai@ayu.ics.keio.ac.jp

ABSTRACT
Sensor database systems need to provide both temporal consistencies of data and persistence to the incoming sensor data streams. For persistence, disk based logging methods have been widely used, however they are not applicable for this purpose because they are awfully tardy. In this paper, we propose the Temporal Consistency Conscious-Write Ahead Logging(T-WAL). The T-WAL is a memory logging method and is a refinement of the Neighbor-Write Ahead Logging(N-WAL). To accelerate logging speed, the T-WAL utilizes temporal consistency constraints to transfer some records together, reveals under looser protocol than the two phase commit protocol and uses private network for logging to avoid congestion. As the looser protocol weakens the guarantee of logging successes, we incorporate a repair system and a checker system to complement the degree of the guarantee. Experiment results showed that the throughput of the T-WAL is approximately 2.05 times greater than the N-WAL when the number of concurrent sensor data streams is 225.

KEY WORDS
Sensor Data Streams, Temporal Consistency of Sensor Data, Data Persistence, Temporal Consistency Conscious WAL

1. Introduction

Recently, a variety of sensor data stream processing applications have appeared. The examples of such applications are real-time financial analysis systems, location information services[1] and intelligent robots[2]. The real-time financial analysis systems collect the current information about markets and analyze the situation of the market with mathematical methods or large amount of stored past data. The location information services manage continuously updating position information of enormous mobile objects such as cars or humans, retrieve services around the objects and provide them in real-time. The intelligent robots recognize the current outside situation by analyzing multiple sensor data streams to communicate with humans.

Though these systems seem to be different, their internal processings are similar. The processings can be divided into two stages. On the first stage, they collect incoming sensor data streams and try to recognize the current context of the physical world. And in the second stage, according to the recognition they select appropriate actions and execute them. Depending on the purposes of applications, processings in the second stages are different. However processings in the first stages are same. All of them recognize the current physical world by monitoring sensor data streams.

One of the most important points on the first stages is the temporal validities of sensor data streams, the freshness and the synchronousness. Because if the sensor data are stale on the first stage, the applications would select inappropriate actions in the sense of time. Therefore it can spoil all of the results acquired in the second stage. Consequently, temporal validities are vital for all of the sensor data stream processing applications.

To monitor streams, incoming streams must be stored into a database system. To enhance the temporal validities, time for insertion processings should be reduced as short as possible. On the other hand, to become data persistent is necessary. This hinders reducing the time for insertion. To make data persistent, an operation to a database must be written into persistent devices. Conventionally disks are used as the persistent device, and the access to a disk is awfully tardy. If persistence can be ignored, the time for insertions can be reduced dramatically. However, needless to say, persistence is mandatory for data in a database.

The tardiness is inevitable as long as disks are used. To overcome the limit, the **Neighbor-Write Ahead Logging(N-WAL)**, that write operations into remote memories before modifying a database, have incorporated in the ClustRa main memory distributed database system[3]. With the N-WAL, log records are transfered to two remote logger sites based on the two phase commit protocol[4]. The N-WAL dramatically decreases the logging time providing almost the same persistence that the WAL provides[5].

The N-WAL can be improved by 1)using private network for logging, relaxing transferring protocol, and 2)considering the temporal validities of the sensor data streams. In this paper, we propose the **Temporal Consistency Conscious WAL(T-WAL)**. The T-WAL never insist on receiving the acknowledgments from the log-

gers. This policy degrades the confidence of successes of log transfers. To complement the confidence, we have adopted two approaches. First, by appending a serial number to each log record, we confirm the arrival of each log record. And second, by checking the existence of a part of log records on the loggers we enhance the confidence of log records. The checkers mostly select records that really exist, but sometimes select records that does not exist to partially deal with the Byzantine failures[6]. Using the same network for both sensor data streams and logging causes traffic congestion and that leads to performance degradation. So we divide the network to avoid the congestion. We give to database system site two network interfaces. One interface is used to process incoming sensor data streams and the other is dedicated for the logging. And the use of multi-cast further enhances logging speed by reducing the number of the log transfers.

This paper is organized as follows. Section 2 describes a related work. In section 3 we present the T-WAL and we evaluate its performance in section 4. Section 5 contains conclusion.

2. Related work: Neighbor-Write Ahead Logging

It is difficult to enhance logging speed as long as disks are used, because a disk access is awfully tardy and this cannot be resolved for the sake of its machinery nature. The use of memories for logging dramatically enhances the logging speed compared with the use of disks. This technique has been incorporated into the ClustRa distributed main memory database system[3]. To the best of our knowledge, the ClustRa is the only database system that incorporates memories for logging. This technique was named the **Neighbor-Write Ahead Logging(N-WAL)**.

2.1 Architecture and protocol

We show the architecture of the N-WAL in the figure 1. A database system and two loggers stay on different sites. The database system and the loggers communicate with each other using the same network. Though the original work[3] used the ATM for the network, the protocol can be applied for the Ethernet. And in this paper we describe the N-WAL on the Ethernet.

On the N-WAL, log records are transfered to two logger sites using two phase commit protocol[7][4]. For sensor data streams, the N-WAL protocol reveals as follows.

1. A database backend receives a new sensor data from a sensor process and appends a timestamp and a log sequence number to the data.

2. The database backend asks both loggers whether they can execute logging or not.

3. The database backend receives answers from the both loggers.

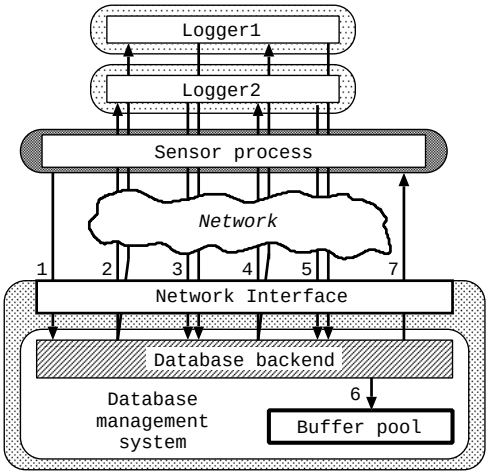

Figure 1. The N-WAL architecture

4. If both of the answers are “OK”, then the database backend sends a log record to the two loggers. Otherwise the database backend aborts the insertion.

5. The database backend waits until the both loggers reply answers.

6. After receiving both answers, the database backend inserts the new data into the buffer pool.

7. Finally, the database backend sends a commit to the sensor process.

2.2 Problems of the N-WAL

We think there are two points that can be improved in the N-WAL.

1. Considering the natures of applications

   The N-WAL is not optimized for continuously arriving sensor data streams. By considering the natures of the monitoring applications, the performance of the logging can be enhanced.

2. Protocol and architecture

   The N-WAL can use more than two loggers on different sites from a database system site. However, the N-WAL reveals under the two phase commit protocol, and the protocol insists on replying from all of the loggers. Therefore, as the number of loggers increases, the number of replies that a database system have to receive also increases, and that incurs traffic congestion. The congestion will probably degrades the logging speed.

   The N-WAL uses the same network for both loggings and sensor data streams. Since sensor data streams continuously arrive, loggings must be executed continuously. Hence the congestion would continuously occur. The use of the same network is inappropriate for the N-WAL when dealing with sensor data streams.

3. Temporal Consistency Conscious-Write Ahead Logging

In this section, we present the new WAL technique, **Temporal Consistency Conscious- Write Ahead Logging(T-WAL)** that enhances logging speed solving the problems described in the last of the section 2. To accelerate logging speed, the T-WAL utilizes temporal consistency constraints, reveals under relaxed protocol than two phase commit protocol and uses private network for logging.

3.1 Using temporal consistency constraints of the monitoring applications

The applications that monitor sensor data streams need temporally correct sensor data. The correctness is twofold. They are referred as the freshness and the synchronousness[8]. The freshness is related to only single sensor data stream and the synchronousness is related to all of the sensor data streams. In this paper, we define them as follows.

- Freshness

  Difference between current time and the arrival time of the last sensor data of a sensor data stream that is monitored by an application.

- Synchronousness

  Maximum difference among the arrival times of last sensor data in all of the streams that are monitored by the application.

If a monitor application periodically reads the freshest sensor data and the period of sensor data arrivals is faster than the period of the monitor, the number of meaningful data is at most one. That is the last sensor data arrived just before the monitor reads. All of the other data are meaningless because they are not read.

The T-WAL utilize this nature of monitoring applications to accelerate insertion processing. The T-WAL sends a set of log records and appends sensor data in a lump. This accelerates insertion processing. But if the number of records of the log record set is too large, then the insertion might be done after reads of monitoring applications. To avoid this undesirable situation, we adjust the number of the log record set. We refer to the number of log records in the log record set as the **NUMLOG**. The **NUMLOG** is calculated as follows.

1. In the case that freshness and synchronousness are not required:

   Define **I_PERIOD** as the insert period of a sensor data stream;
   Define **R_PERIOD** as the read period of a monitoring application;
   Define **MAXLOG** as
   R_PERIOD / I_PERIOD;

   In this case, the **NUMLOG** is calculated as:

   $$1 \leq \mathbf{NUMLOG} \leq \mathbf{MAXLOG},$$

   And **MAXLOG** must be divided by **NUMLOG** to prevent expanding the difference between the time the monitoring application reads data and the time the data are inserted.

2. In the case that freshness and synchronousness are required:

   Define **I_PERIOD** as the insert period of insertions;
   Define **R_PERIOD** as the read period of a monitoring application;
   Define **MAXLOG** as
   R_PERIOD / I_PERIOD;

   Define **FRESH** as the freshness of a monitoring application;
   Define **SYNCH** as the synchronousness of a monitoring application;
   #Both **FRESH** and **SYNCH** are defined larger than **I_PERIOD**;
   Define **TEMP_CONS** as the temporal consistency constraint, that is the smaller one between **FRESH** and **SYNCH**;

   In this case, the **NUMLOG** is calculated as:

   $$1 \leq \mathbf{NUMLOG} \leq \mathbf{MAXLOG},$$

   And **MAXLOG** must is divided by both **NUMLOG** and **TEMP_CONS** to prevent expanding the difference between the time the monitor reads data and the time the data are inserted, and to execute insertions within temporal consistencies before the time the monitoring application reads data.

3.2 Architecture and protocol

The T-WAL uses two network interfaces. One is dedicated for the execution of the logging and the other is used to receive sensor streams and any other queries. This approach avoids the congestion incurred by the collision of the sensor streams and the log records. We adopted UDP/IP multi-cast for sending log records from the database system to the loggers to reduce transmission cost. With the multi-cast, the database needs to send a log record only once. If the unicast is used instead of the multi-cast, the database system has to send the same

packet the number of loggers. And the database system never make loggers reply acknowledgments to the database system. The packet transmission from logger sites occur only when the loggers found out the omission of the log records.

We show the architecture of the T-WAL in the figure 2. The T-WAL reveals as follows.

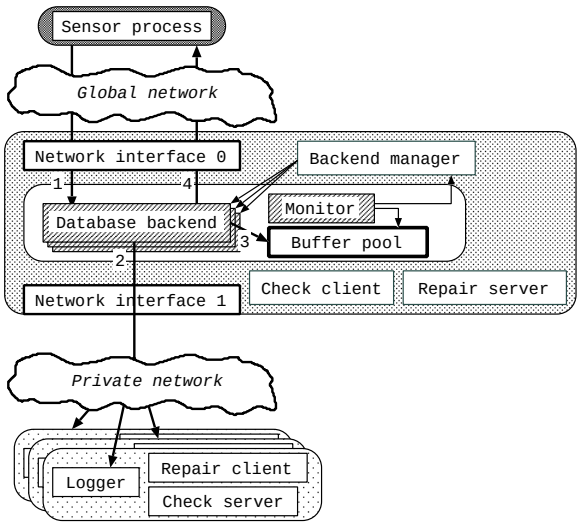

Figure 2. The T-WAL architecture

1. A database system receives a sensor update from a sensor process.
2. If the number of log records is equal to the **NUMLOG**, the database system appends a time stamp and a log sequence number to the updated data and sends the log record to several loggers with the UDP/IP multi-cast.
3. The database system appends the new sensor data into the buffer pool.
4. The database system sends a commit to the sensor process.

The number of steps required for the T-WAL is three steps less than the N-WAL. This also leads the speed up for logging.

The architecture for the T-WAL consists of four components. They are the database system, the logger, the repair system and the check system. We describe them in the following.

- Database system

  The database system consists of query processor, concurrency management, index management, buffer management and recovery management. The query processor only deals with SELECT, INSERT, CREATE, DROP. The concurrency control protocol is the multi-version concurrency control[9]. This is natural because the sensor data streams can be regarded as multi-version records. Index policy is hash. Buffer manager uses FIFO for the sensor data streams and asynchronously transfers the sensor data set into the disk when the buffer is filled. As during the transferring the buffer is write locked, the sensor updates are blocked and hence performance degradation occurs. To avoid this undesired situation, the database system have two buffers for each sensor data stream and avoid the blocking state by switching them.

- Logger

  A logger receives log records from a database system. The logger stores log records in a buffer pool on the local memory until the buffer is fully satisfied. When the buffer is fully satisfied, the logger writes all of the log records in the buffer into the disk in a lump. To avoid blocking, the logger uses two buffers for each sensor data stream. This strategy is the same as the database system's one described above. Loggers receive log records on the UDP/IP multi-cast.

- Repair system

  As shown in the figure 2, the T-WAL never insist replying acknowledgements. Though this is for the speed, it is dangerous not to receive acknowledgments from loggers. To become the T-WAL protocol safe, the loggers use the log serial number for confirmation. If the log serial number of a new log record is not next to the previous one, the logger asks the repair client on the same site to bring the log record.

  The repair server on the database site searches the data, creates a log and returns it. The connection between the repair client and the repair server is the TCP/IP. TCP/IP is slow but reliable. Since the repair can be executed asynchronously, we adopted the TCP/IP.

- Check systems

  To enhance reliability of the log records, the checker client on the database system site periodically asks checker servers the existence of randomly selected particular log records. The checker server inspects whether the log records exist or not. The period of checks is related to the size of buffers in the loggers and this inspection is done before the log records are written into the disk.

  This checking approach can know at least whether the loggers stop or not, but does not know the loggers work normally or not. In addition to that, it partially deals with the Byzantine failures[6], the checker asks whether the loggers contains not only true data, but also false data. Only when all of the checks are correct, the checker clients regard that the logger works normally. The connection between the checker client and the checker servers is TCP/IP.

- Monitors

  Monitors are created from the backend. At first, they notify their temporal consistencies to the backend manager. And they periodically watch sensor data streams in the shared memory.

- Backend manager

  The backend manager manages the temporal consistencies of all of the monitors and notify them the backends when a new monitor is created or a monitor is dropped.

4. Evaluation

We show the performance of the T-WAL through the experiment in this section. We describe network configurations, machine environments, experimental contents and the result.

4.1 Machine features

We show the features of the machine for the database system in the table 1, the machine for the loggers in the table 2 and the machine for the sensor data stream generating process in the table 3, respectively. All of the network interfaces are 100 Mb fast Ethernet and the number of CPU is one.

Table 1. Machine features of the database system

Component	Description
OS	Linux Kernel 2.4.18
Disk	IBM-DPTA-372050, ATA DISK drive
CPU & Memory	Pentium III 600MHz, 128MB

Table 2. Machine features of the loggers

Component	Description
OS	FreeBSD 4.4 RELEASE
Disk	IBM-DTTA-350640
CPU & Memory	Pentium II 300MHz, 64MB

Table 3. Machine features of the sensor processes

Component	Description
OS	Linux Kernel 2.4.18
Disk	IBM-DTTA-350640
CPU & Memory	Pentium III 1GHz, 256MB

4.2 Experimental result

To evaluate the T-WAL, we measured the time for concurrent sensor data streams. Each sensor data stream was consisted of 5000 updates. We measured the response time for each sensor data insertion and the throughput of the sensor data insertions. In the experiments, the NUMLOG was set to 1, 4 and 128 and the number of concurrent sensor data streams was set to 25, 50, 75, 100, 125, 150, 175, 200 and 225. The result of response time is shown in the figure 3 and the result of throughput is shown in the figure 4.

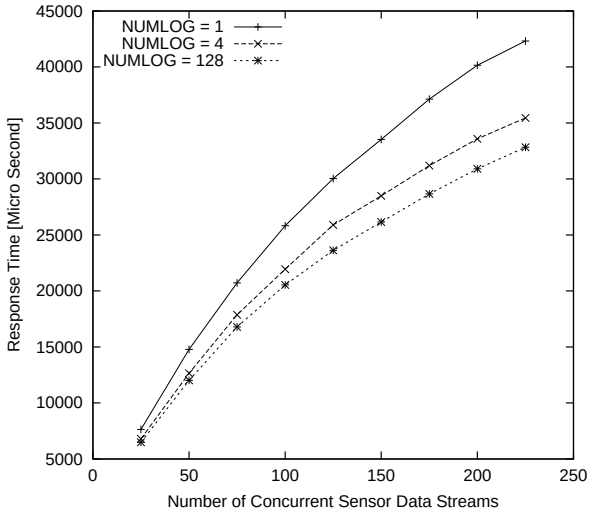

Figure 3. Response time

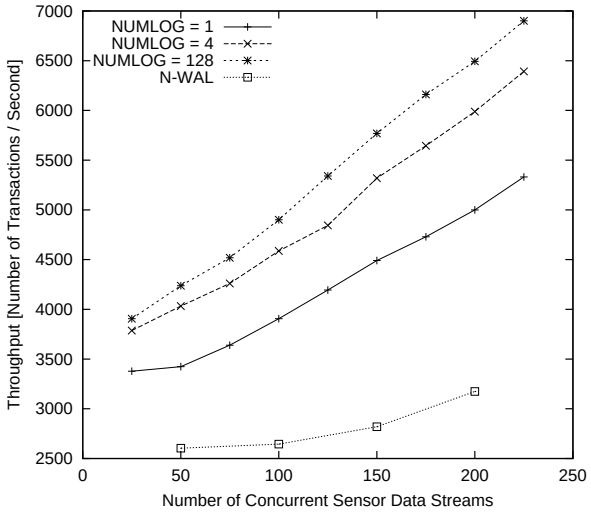

Figure 4. Throughput

The figure 3 shows that as the NUMLOG increases, the response time becomes shorter. When the number of concurrent sensor data streams is 225, the NUMLOG-4 is approximately 19.4% faster than the NUMLOG-1 and the NUMLOG-128 is approximately 28.9% faster than the NUMLOG-1. In all of cases, the response time increases less than linear line. This implies that the T-WAL is applicable to a lot concurrent sensor data streams.

The figure 4 shows that as the NUMLOG increases, the throughput increases. When the number of concurrent sensor data streams is 225, the throughput of the NUMLOG-4 is approximately 19.9% greater than the the throughput of the NUMLOG-1, and the throughput of the NUMLOG-128 is approximately 29.5% greater than

the throughput of the NUMLOG-1. When the number of concurrent sensor data streams is 200, the throughput of the NUMLOG-128 is approximately 2.05 times greater than the N-WAL. This result clearly shows the superiority of the T-WAL.

5. Discussion

In this section, we discuss the confidence and the speed about the T-WAL.

- The Hi-speed network and non volatile memories

  In this research, we used the Ethernet as the network. If we can use faster network like the RHiNET[10], the performance will probably enhance. Since the speed of network has been increasing, the effectiveness of the T-WAL would be enhanced in the near future. However, recently non-volatile memories like MRAMs have appeared. If this kind of memories become popular, the T-WAL will finish its mission and vanish because that device makes the local memories persistent.

- Dynamic adjustment of the NUMLOG

  In this paper, we assumed the periods of sensor data streams are constant. However in fact, sensor data streams would arrive at a database system with some error from constant period. To deal with this problem, dynamically adjustment of the NUMLOG by investigating whether monitors satisfy or not.

6. Conclusion

Sensor database systems need to provide both temporally consistent data and persistence to the incoming sensor data streams. For persistence, disk based logging methods have been widely used, however they are not applicable for this purpose because because of the tardiness. For that We proposed the temporal consistency conscious-WAL(T-WAL) in this paper. The T-WAL is a memory logging method and is a refinement of the neighbor-WAL(N-WAL). The T-WAL needs two network interfaces , applies looser protocol than two phase commit protocol and utilize the monitors' temporal consistency constraints. As the looser protocol weakens the guarantee of logging successes, we incorporated repair system and checker system to complement the guarantee. Experiments showed that as the number of log records that are logged increases, both response time and throughput improves.

References

[1] Michimune Kohno and Yuichiro Anzai. An Adaptive Sensor Network System for Complex Environments. In *Proceedings of 5th International Conference on Intelligent Autonomous Systems*, pp. 21–28, 1998.

[2] Yuichiro Anzai. Human-Robot-Computer Interaction: A New Paradigm of Research in Robotics. *Advanced Robotics*, Vol. 8, No. 4, pp. 357–369, August 1994.

[3] Svein-Olaf Hvasshovd, Øystein Torbjørnsen, Svein Erik Bratsberg, and Per Holager. The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response. In *Proceedings of 21th International Conference on Very Large Data Bases*, pp. 469–477, September 1995.

[4] Øystein Torbjørnsen, Svein-Olaf Hvasshovd, and Young-Kuk Kim. Towards Real-Time Performance in a Scalable, Continuously Available Telecom DBMS. In *Proceedings of 1st International Workshop on Real-Time Data Bases: Issues and Applications*, pp. 22–29, March 1996.

[5] Øystein Torbjørnsen. *Multi-Site Declustering Strategies for Very High Database Service Availability*. PhD thesis, The Norwegian Institute of Technology, University of Trondheim, 1995.

[6] Nancy A. Lynch. *Distributed Algorithm*. Morgan Kaufmann, 1996.

[7] Jim Gray and Andreas Reuter. *Transaction processing: concepts and techniques*. Morgan Kaufmann, 1993.

[8] Koich Munakata, Masatoshi Yoshikata, and Shunsuke Uemura. Integrating Periodic Data Sequences Based on Freshness and Synchronousness. In *Proceedings of the Tenth International Workshop on Research Issues on Data Engineering: Middleware for Mobile Business Applications and E-Commerce*, pp. 47–54, 2000.

[9] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 2001.

[10] Hiroaki Nishi, Koji Tasho, Tomohiro Kudoh, Junji Yamamoto, and Hideharu Amano. RHiNET-1/SW: an LSI switch for a local area system network. In *International Symposium on Low-Power and High-Speed Chips, COOL Chips III*, pp. 175–187, April 2000.