Design for the Distributed Data Locator Service for Multi-site Data Repositories

H. Nakanishi[a,c,*], K. Yamanaka[b,c], S. Tokunaga[d], T. Ozeki[d], Y. Homma[d], H. Ohtsu[d], Y. Ishii[d], N. Nakajima[a], T. Yamamoto[a], M. Emoto[a], M. Ohsuna[a], T. Ito[a], S. Imazu[a], M. Nonomura[a], M. Yoshida[a], H. Ogawa[a], H. Maeno[a], M. Aoyagi[a], M. Yokota[a], T. Inoue[a], O. Nakamura[a], S. Abe[b,c], S. Urushidani[b,c]

[a]*National Institute for Fusion Science, NINS, Toki, Gifu 509-5292, Japan.*
[b]*National Institute of Informatics, Chiyoda-ku, Tokyo 101-8430, Japan.*
[c]*The Graduate University for Advanced Studies, SOKENDAI, Hayama, Kanagawa 240-0193, Japan.*
[d]*National Institutes for Quantum and Radiological Science and Technology, Rokkasho, Aomori 039-3212, Japan.*

Abstract

The Remote Experimentation Centre (REC) in Japan has been preparing to replicate the full dataset of ITER over 10 000 km distance. In such a multi-site data repository environment, the data location informing service will be essential to find and retrieve the data efficiently. Considering the long latency time and the self sustainability of remote sites, the data location database should be running at each repository site. Multi-master asynchronous replication between cooperating databases will be essential to realize the remote experimental collaborations in fusion research. This study has investigated the functional differences of some relational databases and found that Postgres BDR has the expected database replication capabilities. Bi-directional replication (BDR) tests by using the LHD database and SNET revealed that the throughputs are sufficient for remote collaborations in fusion experiments.

Keywords: multi-site data repository, data locator service, bi-directional replication, Postgres BDR, ITER REC, LHD, SNET

1. Introduction

In modern fusion experiments, remote data access has already come into wide use in both domestic and international research collaborations. The SNET fusion data exchanging platform in Japan interconnects four fusion experimental sites, LHD, QUEST, GAMMA10, and TST-2, over 1 000 km distance [1]. SNET enables remote collaborators to seamlessly access each site's data as if they were in a local site.

Similarly, the Remote Experimentation Centre (REC) in Rokkasho, Japan is planning to replicate the full dataset of ITER data over 10 000 km distance, where high-performance computing resources are ready for off-site analyses on ITER physics data [2]. Prior to this study, our group has performed some series of inter-continental massive data replication tests between ITER and the REC sites [3, 4, 5].

In such multi-site data repository environments, the data location informing "locator" service will be essential for finding the best data server from which users can retrieve the data most efficiently. Considering that the latency time is more than 100 milliseconds for inter-continental network transactions, not only the data repositories but also the locator servers should be distributed to multiple sites. Since the data location information will be stored and served by means of a relational database (RDB), such as PostgreSQL or MySQL, realtime synchronization between the distributed RDBs will be necessary to provide a consistent data locator service around the world.

From years' operational experience of the SNET remote sites, we also found that the operational independence of the remote site is quite important against unexpected and planned service outages of the original site. A typical example is the annual electricity inspection by law with an power outage of the whole site, which never and should not concern the remote site work continuity such as data accesses for analyses.

In this study, bi-directional replications between multi-master locator RDBs has been tested by using the LHD data system and SNET. Since the current structure of the SNET distributed data system adopts a master-slave RDB replication architecture, the system structure must be redesigned for applying the multi-master bi-directional replication. Such structural refinements for the distributed databases would be a common issue not only for SNET's multiple sites but also for the bi-directional replication between ITER and the REC.

In addition, some performance tests have been performed by using the PostgreSQL version 9.4 with the extension of bi-directional replication (BDR) [6]. It is because PostgreSQL has been selected as the standard RDB by ITER CODAC [7]. Before the actual performance test, some technical surveys on other RDB replication methods have been made with some comparisons and discussions.

2. LHD Data System and SNET Platform

After almost twenty years of high-temperature hydrogen plasma experiments, the Large Helical Device (LHD) has successfully started the deuterium experiment since March 2017, in which further plasma performance improvement is envisaged to contribute to the fusion reactor design [8].

*Corresponding author; phone: +81 572 58 2232,
Email address: `nakanishi.hideya@nifs.ac.jp` (H. Nakanishi)

As for the data system of LHD, the recommend-type facilitator model has been adopted [9], in which two independent transactions are required in every data access: At first the data client makes an inquiry to the relational database to know the necessary data location(s), and secondly the client requests to retrieve data from the previously identified data server(s). Those two steps use different protocols. The former uses an SQL-based database query, and the latter uses an FTP-like homemade protocol. Such a separation has been adopted to avoid any bottlenecks for handling a large number of simultaneous data users and also massive-sized data inputs and outputs at the same time.

The LHD data system once adopted a distributed key-value store (KVS) which is a kind of the so-called NoSQL database. Even though it had many advantageous features, we sometimes suffered from operational troubles especially in recovering from some node failures [10]. Such the lack of the system durability in recovering from fault conditions let us conclude to abandon it and continue using a SQL-based RDB instead because we have never experienced any troubles at all for more than 20 years PostgreSQL operation in LHD.

Together with three universities employing fusion experimental devices, the LHD data system has been organizing the Fusion Virtual Laboratory (FVL) in Japan since 2008 [11]. At the present time, the participating sites are LHD at NIFS, QUEST at Kyushu University, GAMMA10/PDX at University of Tsukuba, and TST-2 at the University of Tokyo, connecting through the layer-2 and layer-3 virtual private network (VPN) of SNET [12].

In such a multi-site data production and consuming environment, we sometimes suffered from the accidental loss of network connectivity. Under the present system structure shown in Fig. 1 (left), the acquired raw data can be queued within the DAN archiver while waiting for the network connectivity to be restored thereby allowing the raw data to be sent to the primary data server. However, it is difficult for the connection-lost site users to make an inquiry to search the data locations because the proxy server cannot relay requests if the primary indexing DB or intermediate network is down. Likewise, new data registration is not possible thus that they cannot be found by any other data consumers.

The LHD data storage adopts the multiple stages: i) on DAQ node, ii) SSD array, iii) HDD raid cluster, and iv) Blu-ray Disc library, in which the archiving data can be queued and served at every stage. However, the indexing DB has only master-slave replication locally for the data safety.

Consequently, the ideal condition for such multi-site data sharing cases can be considered that the remote site should continue the data operation independently from the primary data archiving repository and the indexing DB even while the network connectivity was lost. Of course, the queued data and the modified location information should be automatically re-synchronized to the primary data archiving repository and to the indexing DB as soon as possible when the network connectivity is back.

To satisfy the above mentioned requirements, the multi-master database synchronization should be applied for the SNET distributed data system. Figure 1 (right) shows the schematic diagram of the structural refinement. Being different from the proxy-based remote query-forwarding service, this multi-master databases can accept the local read/write transactions even while an accidental loss of network connectivity happens at the remote site. While network connectivity lost, the data retrieving clients can look for the data locations on the replicated indexing DB and read the queued data at the data acquisition and archiving server (DAN Archiver) within the site.

As for the re-synchronization mechanism between primary and replicated indexing DBs, technical discussions will be given in the later section. However, it must be noted that all the data producers of the fusion experimental device, including the human data analysts, will generate their own data and register them with a unique name given beforehand. Most of the physics data will be also managed along with the experimental pulse number or date, and thus, generated data will be simply accumulated without any modification afterward.

Therefore, if the data registration/modification privileges are properly managed, there could not be any transaction conflict in newly registering or modifying the data records. This fact means that the data synchronization timing and the conflicting transaction management are not so critical for the data indexing databases of the fusion device and plant operations.

3. ITER UDA and Replicated Repository

Under the Broader Approach (BA) activities, the ITER Remote Experimentation Centre (REC) has been preparing to make a full replication of ITER data at Rokkasho, Japan [13]. ITER data system and its access methods are named as ITERDB and the Unified Data Access (UDA), respectively [14, 15, 16]. ITER UDA adopts the broker-type facilitator model [9], in which all the client/server communications will be carried through the UDA server. Figure 2 (left) shows the schematic diagram [17].

As the ITER data archiving repository is the original full set of ITER data, not only the bulk dataset but also the indexing DB would be a single point of failure (SPOF). Long-distance accesses to the only one indexing DB may involve a large risk for the remote users suffering from any accidental losses of the network connectivity.

The SPOF problem of the indexing DB should be solved by introducing the service and server redundancy. If having the replicated indexing DBs at the remote sites, each of them might improve the site independence because it can accept remote users' inquiries for the data locator service even when the primary indexing DB would have an accident.

In fusion experimental data analyses, the requirement conditions for the replicated indexing DBs considering the off-site data users can be thought of as follows:

(i) Service continuity, no coincident with any stops of primary indexing DB
(ii) Service suspension at any time and the subsequent recovery of synchronization, not only for system failures but also for planned maintenance or site power outage, etc.

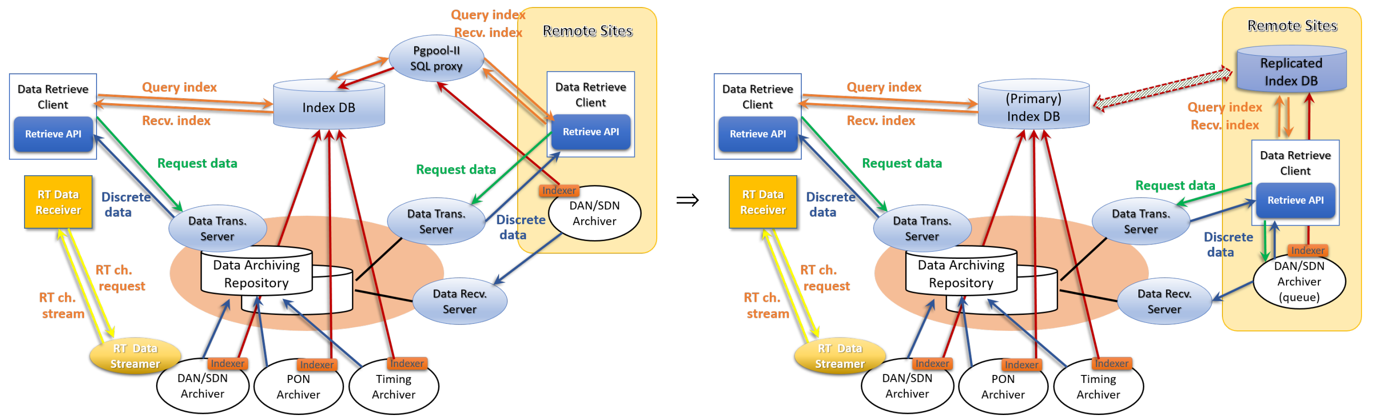

Figure 1: LHD data system only with a primary indexing DB (left) and the multi-site extension having bi-directionally replicated indexing DBs at each remote site (right): The SQL proxy server can forward the SQL query commands and their returned results to/from the primary indexing DB. Not only DAN but also some other synchronous data (SDN) and plant operation data (PON) archivers migrate and register their data to the repository. Real-time data monitoring streams are independently transfered between their own client and server.

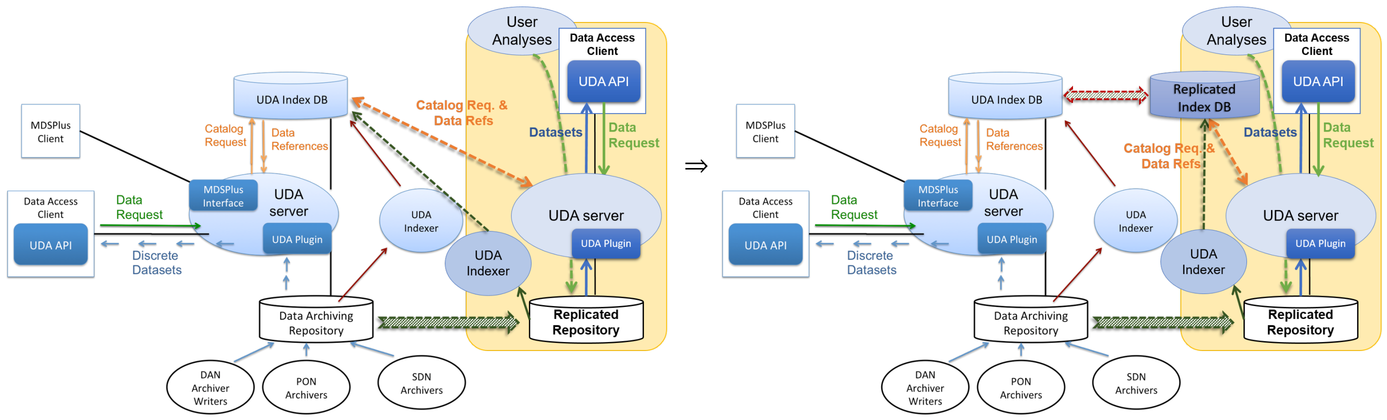

Figure 2: ITER UDA structure and the replicated repository: The yellow-hatched areas for remote sites are added by us upon the original UDA design of white background (left) [17]. If the there is only one indexing DB, it would be a single point of failure (SPOF). Remote site's independence against accidental loss of long-distance network connectivity or planned power outages can be improved if the replicated indexing DB could continue operation independently from the primary indexing DB even though the realtime synchronization is temporarily lost (right). The indexer process should always register a new data entry synchronously with the data migration process making a new copy of the data or moving to other place, locally or remotely.

(iii) Delayed replication ability, against any loss or temporary failure of intermediate network links.

There are primarily two choices for the way to synchronize the DB records between primary and replicated DBs: the master-and-slave(s) or the multiple-masters structure. Because no slave DBs can continue to accept the write transactions independently from the primary DB, it is expected to adopt multiple-master formation between ITER and the replicated indexing DBs of the remote sites. A typical example of the conceivable structure between ITER primary and the REC replicated data archiving repositories is shown in Fig. 2 (right).

As for the data archiving repository itself, automatically generated ITER primary data should be promptly replicated to the remote site(s) according to the pulse sequences. On the other hand, secondary data products such as analyses results made outside IO could be generated independently from the primary site operation. Not for disturbing the off-site data analyses, the replicated repository should accept new data registrations for those secondary data products. Those data should be also served or shared together with the indexing DB for at least the site users, as shown in Fig. 2 (right).

4. RDB Replication Methods

Replication is a technology to improve load balancing and/or fault tolerance capabilities having a replicated "replica" database with the same contents. There are two types of replication: synchronous replication and asynchronous replication [18].

Synchronous replication transfers the update information to all the related master and slave databases, and waits for the completion of all the database updates. Although synchronous replication always guarantees the data consistency between each database, it involves performance degradation in synchronizing every data update.

In asynchronous method, the update information will be at first stored in the master database, and later transferred to the other databases for replication. Asynchronous replication causes less performance degradation. However, there remain some time gaps of data inconsistency until the update propagation is completed. Due to the gap time, information inconsistency would possibly be caused if some conflicting transactions would be executed. Then, some special treatment would be necessary to dissolve the inconsistency. Many asynchronous replication methods adopt rather a simple treatment basis, "Last update wins," in which the transaction executed last in time always rewrites the previous transactions and remains effective thereafter.

As mentioned in the previous section, the multiple-master configuration of the DB replication will be necessary for the internationally collaborating fusion experiment such as ITER. Every physics and engineering data of fusion experiments are generally distinguished by a unique name given beforehand and a unique number given sequentially for each experimental discharge. Therefore, accidental collisions of update information will not occur in principle if data revision management is carried out properly. This is why the asynchronous replication is a realistic solution considering the long-distance distributed DB operation at remote collaboration sites.

In this section, brief overviews will be given for some varieties of replication methods for PostgreSQL relational database [19]. PostgreSQL is a widely used open source software (OSS) having a long history of development by mostly volunteer developers. Therefore, various implementations can be found for its replication methods. Their functional comparisons are shown in Table 1.

4.1. Synchronous Replication Methods

There are two major approaches for the implementation of the synchronous replication between PostgreSQL DBs. One is to use the SQL statement-based middleware which duplicate the received SQL statements to deliver to each related DB and wait for the completion of all the transactions. One well-known problem on this method is that generated results of some SQL functions, such as random() or now(), might be different between multiple DBs.

Another problem is the working location of this middleware. Since such the middleware works as a "proxy" server transferring the SQL statements, it could not run independently without communicating with the backend DB. The configuration is easy to setup and therefore is very popular so that we have applied this method between the LHD data system and the SNET remote sites, as shown in Fig. 1(left). The server software "Pgpool-II" is used as the SQL proxy for the remote SNET sites [21]. However, this configuration does not realize the site independence of the remote sites for the operations of data updates.

Another synchronous replication method is to use the streaming replication with synchronous_commit = on where PostgreSQL waits for the transaction committed on both master and slave databases [20]. Multi-master configuration is not realized by using this method.

Postgres-XC and Postgres-R are the projects for enabling the shared-nothing DB cluster configuration by modifying the original PostgreSQL codes [22, 23]. They apply the two-phase commit capability on PostgreSQL for implementing the multi-master synchronous replication. These tightly coupled clustering databases have different objectives, such as high-availability (HA), failover, parallel transaction processing, load balancing and performance scale-out, comparing to the loosely coupled long-distance DB replication between ITER and the REC.

4.2. Trigger-based Asynchronous Replication

Database trigger is a commonly implemented function which can automatically execute user defined procedures awaken by every transaction event on the specific tables. It can execute not only the stored scripts but also external commands through the procedure, and thus, is often used for implementing the processes running to monitor and transfer the DB changes for replication. As the replication will be made asynchronously, this solution involves a risk of data loss in case the primary server would have an accidental stop.

There are some popular software packages for master-slave and multi-master replications, such as Slony-I and Bucardo.

Slony-I can make single-master multi-slave DB replication asynchronously. Its replication is triggered by INSERT, UPDATE, DELETE, TRUNCATE on the target table, and the update log is transferred to the slave DBs. Setup for Slony-I replication is somewhat complicated because both side's applications require different but mutually dependent settings. Londiste is another row-based implementation similar to Slony-I [24].

Bucardo is a free OSS developed for enabling asynchronous replication on PostgreSQL [25]. By using triggers and queuing in the database, it can provide row-based master-slave and multi-master replications. Background daemon processes written in Perl programming language are needed to transfer the transaction data and replicate them to the related DBs. For multi-master configurations, standard or customized method can be selected for resolving the transaction conflicts. It cannot handle SQL commands on data definition language (DDL), such as CREATE, DROP, or ALTER, nor the large objects because PostgreSQL has no triggers on them.

In both Slony-I and Bucardo cases, a trigger processing daemon should be running which should be called externally from the master database service. Such an independent service daemon could be another SPOF in the DB replication system, and also increase some computing overheads of external process calls, compared to the streaming replication method mentioned below.

4.3. Streaming Replication (Physical Replication)

Streaming replication is implemented by applying the transaction log shipping mechanism. At the master server, modifications on the database are logged to the binary write-ahead log (WAL), which is transferred to the slave DBs and restored via WAL recovery method. This scheme regenerates completely the same database at every slave DB server so that it is often called a "physical replication."

PostgreSQL 9.0 and higher versions enable the streaming replication from a single master to multiple slave DBs. Both

Table 1: Replication types and related packages for PostgreSQL [20]. PG means a build-in function of PostgreSQL.

based-on	sync/async	m-s/m-m	software solution	note
SQL statement	sync	multi-master	Pgpool-II	may cause inconsistent results
DB trigger	async	master-slave	Slony-I, Londiste	needs external daemon process
〃	〃	both	Bucardo	〃
WAL shipping	both	master-slave	PG streaming replication	WAL is a binary block
logical decoding	both	both	PG logical replication	table and row-based
〃	〃	〃	Postgres BDR	〃 + "Last-update-wins"

synchronous and asynchronous replications have been realized, even though some performance degradation may occur in synchronous replications [20].

This method has some restrictions that the replication can only be made on a per-database basis, not on a per-table or per-row basis. Also, the replicated slave DB servers cannot process any data updating transactions. Streaming replication would be rather effective for the use in failover DB cluster configurations.

4.4. Logical Replication

Logical replication is based on similar mechanism to the streaming replication. At the WAL sender on the publisher server, binary WALs are logically decoded from binary blocks to higher-level data manipulations on tables and rows so that they are transferred to the WAL receiver to be applied on the subscriber DB server. This is why the stored results of the SQL commands "random" and "current_timestamp" can be consistent.

Basically, logical replication allows updating transactions to flow to multiple recipients without fixing the role of a specific server as a master or slave. Accordingly, the subscriber databases can accept and process updating transactions. Therefore, there is no apparent way to find the transaction conflict.

Logical replication is still under active development on the most recent version of PostgreSQL. Some expected functionalities are not implemented yet, such as replicating the DDL commands [20]. Postgres BDR is one of the practical implementations of PostgreSQL logical replication.

4.5. Postgres BDR

In case of multiple data repositories and/or data production sites, distributed data locator service will be essential to manage the data indexes locally and also exchange their information with remote sites. To establish the bi-directional database replication is the key technology to enable a research collaboration over far remote sites.

PostgreSQL 9.4 and higher versions have been equipped with the multi-master bi-directional replication (BDR) [6]. Postgres BDR adopts a loosely coupled shared-nothing multi-master architecture. Since BDR is implemented as an extension module of PostgreSQL 9.4 and the higher, it can be loaded to PostgreSQL just by "CREATE EXTENSION bdr" command. As the BDR is based on the PostgreSQL logical replication, data will be replicated on "row-based" rule which is neither by higher-level "statement-based" nor by "log-shipping" algorithm.

One of the most advantageous points of Postgres BDR is that it does not require other external processes than the standard PostgreSQL services so that all the necessary configurations can be written in the PostgreSQL setup files or stored in the embedded database schema. It is quite easy and straightforward to establish multi-master bi-directional replication.

The updating data will be once stored in the local database, and also queued for asynchronous replication, which is not concerned with whether the connection to the remote database is healthy or not. Replication of the queued data would be restarted automatically when the connection is recovered. Such loosely coupled databases are very appropriate for the requirements of ITER-REC and SNET multi-site collaborations.

Postgres BDR has many favorable features, as mentioned. However, there still remain some technical constraints. Some DDL commands, such as CREATE, DROP, and ALTER, are still not supported. A primary key or keyset should be definitely given for every record on tables, which should not be the standard object identifier (OID) on each record. To cope with the transaction conflicts, BDR adopts a simple "Last update wins" rule.

To maintain the favorable replication throughput, the replication interval is expected to be 2 seconds as default. Such a quasi-realtime synchronization does not fit for the time-critical applications. However, it adequately satisfies the requirements for the data replication of fusion experiments.

The performance test results by using the LHD indexing database are shown below.

4.6. Replication of other RDBMS

Oracle database is a well-known commercial RDB management system, which has long been known to provide single-master or multi-master synchronous and asynchronous replications called "Advanced Replication." However, for Oracle Database 12c Release 2 (12.2) and later releases, replication functionalities have been separately provided by the Oracle GoldenGate which enables replications between different versions and different RDB products [26].

Oracle GoldenGate can be understood as a kind of log-shipping streaming replication from the master database to the slave one(s). However, it can setup the streaming replication in the opposite direction in parallel. At the moment, Oracle GoldenGate has a restriction that PostgreSQL cannot be a primary DB to be replicated. This restriction may cause a difficulty for some fusion experimental projects, such as ITER and LHD, which already adopted PostgreSQL as the standard RDB.

Table 2: Performance results on Postgres BDR: Each record size is rather small, practically less than 1 kB. Bold types show the BDR's elapsed time per record.

DB (# of tables)	SQL	# of records	elapsed time /s	time per record /s
ex_note (5)	insert (bdr)	144772	243.6	**16.8×10^{-4}**
″	copy (bdr)	″	23.1	**1.60×10^{-4}**
″	″ (local)	″	0.978	6.76×10^{-6}
setup (167)	copy (bdr)	11557821	971.6	**0.84×10^{-4}**
″	″ (local)	″	62.15	5.37×10^{-6}
index (22)	copy (bdr)	207911053	35654	**1.72×10^{-4}**
″	″ (local)	237544798	581.5	2.45×10^{-6}

MySQL and its derivative MariaDB are also very popular free RDBMS. On their own, MySQL servers provide master-slave replication capabilities based on the binary log shipping method. Not only asynchronous replication but also semi-synchronous replication are possible for faster failover and crash recovery on HA systems [27].

MySQL Cluster enables active-active multi-master replication for mission-critical HA systems with performance scalability [28]. For the cluster configuration, it adopts different data engines and control schemes than the standard MySQL, where more than three nodes should be running for the DB operation.

MariaDB Galera Cluster is a similar HA cluster system based on MariaDB [29]. Both the MySQL Cluster and MariaDB Galera Cluster apply the majority rule for avoiding the split-brain problem so that the minority nodes cannot continue the DB operations [30]. In other words, such cluster configurations do not fit for the loosely tied DB replication over the wide-area network.

5. Performance Verification on Postgres BDR

Actual throughputs of Postgres BDR have been investigated between NIFS, Toki and REC, Rokkasho with the following test conditions:

distance	: ~ 1100 km
RTT	: 16.2 ms (RTT = round-trip time)
Ethernet	: Intel X710-DA4, 10 Gbps
motherboard	: ASUS X99-E-10G WS
cpu	: Xeon E5-2650 v4, 2.2 GHz, 12c/24t
memory	: DDR4-2400 ECC RDIMM, 128 (32x4) GB
storage	: Samsung 960 PRO 512GB (NVMe SSD)

Table 2 shows the actual elapsed time observed by our bi-directional replication tests using the major databases of LHD. Here, BDR's default queuing time of 2 seconds (bdr.default_apply_delay = 2000 ms) was used as it is.

We can find that the processing times with BDR are mostly 2-digit longer than those without BDR. However, BDR copy even takes 0.172 ms or less per record. Not the whole table copy but a single insertion command takes approximately 1.68 ms per record, which is still 1/10 smaller than the network round-trip time of 16.2 ms. In case of ITER and the REC, it becomes less than 1 % of about 180 ms round-trip time. These results show that 2-second replication queuing is effective in keeping the slowdown to an acceptable level even under high-latency networks.

Long elapsed time, such as 35 654 seconds shown in Table 2, is to copy or insert numerous records with the BDR relation already established. Postgres BDR only transfers the differential information between BDR DBs, so that initial data importation for a new replica DB should be made locally on the replica DB before starting the BDR relation. Based on Table 2 results, the necessary time to make a new replica DB would be rather 581.5 seconds of local copy, than 35 654 seconds of BDR copy. It is adequately fast for practical uses even considering the number of records being higher in case of ITER.

As a consequence of our tests, we have found that each replication transaction in Postgres BDR takes rather a short time comparing with the network round-trip time between distant sites with RTTs of more than 10 ms like ITER and the REC, LHD and the SNET. Since every network communication cannot overcome the network latency, *i.e.* RTT, bi-directional database replication like Postgres BDR can be of practical use to be applied between multiple remote collaboration sites for some fusion experiments.

6. Conclusion and Future Works

In order to put widely distributed data repositories of practical use for off-site data analyses, the data location informing service should be running at each repository site to sustain the independent operation against any accidental or scheduled stops of other sites. To satisfy the conflicting requirements of both the site independence and the mutual data synchronization, asynchronous multi-master, *i.e.*, bi-directional, replication should be applied between the cooperating relational databases that serve the data location indexes.

Through a functional survey on popular open-source and commercial RDBMS software, we found PostgreSQL to be the most promising solution, at the moment, for such multi-site DB replications between ITER and the REC, and also between LHD and SNET. Since ITER and LHD have already adopted the PostgreSQL as their standard RDBMS, its extension for the bi-directional replication might cause fewer compatibility issues. Other popular RDB products, such as Oracle and MySQL, seem to be aiming for densely coupled cluster configurations and are oriented towards being applied to mission-critical cases rather than being used in collaborative research uses.

To verify the replication performance, the throughputs of Postgres BDR have been measured on SNET by using the LHD indexing databases. The result shows that the replication speeds are adequately fast compared to the network round-trip time. Thus, we can conclude that Postgres BDR is a very practical solution in both functional and performance viewpoints for our multi-site database replication in fusion experiments.

Even though BDR 1.x was the OSS based on the PostgreSQL 9.4, the successor BDR 2.0 on PostgreSQL 9.6 and the current 3.0 on PostgreSQL 11 and 12 are now non-free licensed software, unfortunately [31]. Nevertheless, we still expect further developments on the standard PostgreSQL logical replication because the BDR developer continues the contribution on it [32]. Our investigations on this matter will be also continued in the near future.

We also plan to make another verification test on relaying replication through more than two sites. In such environments, some selection schemes would be necessary to choose the most efficient repository and also the indexing database.

Acknowledgments

This work is performed with the support and under the auspices of the NIFS research programs (NIFS19KKSH006, NIFS19PLHH281, NIFS19ULII001, NIFS19ULHH006). It was also supported by the QST Collaboration Research under the Broader Approach Activity in 2012–2020.

References

[1] H. Nakanishi, M. Ohsuna, M. Kojima, S. Imazu, M. Nonomura, T. Yamamoto, M. Emoto, M. Yoshida, C. Iwata, M. Shoji, Y. Nagayama, K. Kawahata, M. Hasegawa, A. Higashijima, K. Nakamura, Y. Ono, M. Yoshikawa, S. Urushidani, Data acquisition system for steady-state experiments at multiple sites, Nuclear Fusion 51 (11) (2011) 113014.

[2] IFERC, CSC, `https://www.iferc.org/CSC_Scope.html` (2019).

[3] K. Yamanaka, S. Urushidani, H. Nakanishi, T. Yamamoto, Y. Nagayama, A TCP/IP-based constant-bit-rate file transfer protocol and its extension to multipoint data delivery, Fusion Eng. Design 89 (5) (2014) 770–774.

[4] K. Yamanaka, H. Nakanishi, T. Ozeki, S. Abe, S. Urushidani, T. Yamamoto, H. Ohtsu, N. Nakajima, Long distance fast data transfer experiments for the ITER Remote Experiment, Fusion Eng. Design 112 (2016) 1063–1067.

[5] K. Yamanaka, H. Nakanishi, T. Ozeki, N. Nakajima, J. Farthing, G. Manduchi, F. Robin, S. Abe, S. Urushidani, High-performance data transfer for full data replication between ITER and the Remote Experimentation Centre, Fusion Eng. Design 138 (2019) 202–209.

[6] PostgreSQL Global Development Group, Postgres BDR, `http://bdr-project.org/docs/stable/` (2016).

[7] ITER IDM, System Design Description for CODAC, CODAC DDD (ITER_D_6M58M9) (2014).

[8] Y. Takeiri, et al., Extension of the operational regime of the LHD towards a deuterium experiment, Nuclear Fusion 57 (10) (2017) 102023 (10pp).

[9] J. Bacon, Concurrent Systems, Addison-Wesley, Reading, 1993, [Japanese translation: Toppan, Tokyo (1996)].

[10] H. Nakanishi, M. Ohsuna, M. Kojima, S. Imazu, M. Nonomura, M. Emoto, T. Yamamoto, Y. Nagayama, T. Ozeki, N. Nakajima, K. Ida, O. Kaneko, Revised cloud storage structure for light-weight data archiving in LHD, Fusion Eng. Design 89 (5) (2014) 707–711.

[11] H. Nakanishi, M. Kojima, C. Takahashi, M. Ohsuna, S. Imazu, M. Nonomura, M. Hasegawa, M. Yoshikawa, Y. Nagayama, K. Kawahata, Fusion virtual laboratory: The experiments' collaboration platform in Japan, Fusion Eng. Design 87 (12) (2012) 2189–2193.

[12] T. Yamamoto, Y. Nagayama, H. Nakanishi, S. Ishiguro, S. Takami, K. Tsuda, S. Okamura, Configuration of the virtual laboratory for fusion researches in Japan, Fusion Eng. Design 85 (3-4) (2010) 637–636.

[13] J. Farthing, T. Ozeki, S. C. Lorenzo, N. Nakajima, F. Sartori, G. D. Tommasi, G. Manduchi, P. Barbato, A. Rigoni, V. Vitale, G. Giruzzi, M. Mattei, A. Mele, F. Imbeaux, J.-F. Artaud, F. Robin, J. Noe, E. Joffrin, A. Hynes, O. Hemming, M. Wheatley, S. O'hira, S. Ide, Y. Ishii, M. Matsukawa, H. Kubo, T. Totsuka, H. Urano, O. Naito, N. Hayashi, Y. Miyata, M. Namekawa, A. Wakasa, T. Oshima, H. Nakanishi, K. Yamanaka, Status of the ITER remote experimentation centre, Fusion Eng. Design 128 (2018) 158–162.

[14] G. Abla, G. Heber, D. P. Schissel, D. Robinson, L. Abadie, A. Wallander, S. M. Flanagan, ITERDB—The Data Archiving System for ITER, Fusion Eng. Design 89 (5) (2014) 536–541.

[15] R. Castro, Y. Makushok, L. Abadie, S. Pinches, D. G. Muir, J. Hollocombe, J. Vega, J. Faig, G. García, ITER Unified Data Access (UDA), in: 11th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, 8–12 May 2017, Greifswald, Germany, 2017, O-31.

[16] R. Castro, L. Abadie, Y. Makushok, M. Ruiz, D. Sanz, J. Vega, J. Faig, G. Román-Pérez, S. Simrock, P. Makijarvi, Data archiving system implementation in ITER's CODAC Core System, Fusion Eng. Design 96-97 (2015) 751–755.

[17] ITER IDM, UDA architecture diagram, UDA user manual (TPLTKG v2) (2018).

[18] R. Miyazawa, K. Uchida, PostgreSQL replication methods and functional comparisons (in Japanese), `https://www.techscore.com/tech/sql/replication/` (2007).

[19] PostgreSQL, `http://www.postgresql.org/` (2020).

[20] PostgreSQL Global Development Group, PostgreSQL Documentation—Chapter 26. High Availability, Load Balancing, and Replication, `https://www.postgresql.org/docs/12/high-availability.html` (2020).

[21] Pgpool-II, `https://www.pgpool.net/` (2020).

[22] Postgres-XC, `https://postgresxc.fandom.com/wiki/Postgres-XC_Wiki` (2013).

[23] Postgres-R, `http://www.postgres-r.org/` (2010).

[24] Londiste, `https://wiki.postgresql.org/wiki/SkyTools#Londiste` (2020).

[25] Bucardo, `https://bucardo.org/Bucardo/` (2020).

[26] Oracle, GoldenGate, `https://www.oracle.com/middleware/technologies/goldengate.html` (2020).

[27] MySQL, `http://www.mysql.com/`.

[28] Oracle, MySQL Reference Manual—MySQL NDB Cluster, `https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html` (2020).

[29] MariaDB, Galera Cluster, `https://mariadb.com/kb/en/galera-cluster/` (2020).

[30] Wikipedia, Split-brain (computing), `https://en.wikipedia.org/wiki/Split-brain_(computing)` (2020).

[31] 2ndQuadrant, BDR—Advanced Clustering & Scalling for PostgreSQL, `https://www.2ndquadrant.com/en/resources/postgres-bdr-2ndquadran` (2020).

[32] 2ndQuadrant, pglogical — The next generation of logical replication for PostgreSQL, `https://www.2ndquadrant.com/en/resources/pglogical/` (2020).