様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　３ 年　 ５ 月　 ５ 日現在

機関番号：１３９０３
研究種目：基盤研究(C)（一般）
研究期間：2018～2020
課題番号：１８Ｋ１１３２４

研究課題名（和文）マルチレイヤリバースプロキシによる分散キーバリューストアへのトランザクション実装

研究課題名（英文）Transparent Transaction Processing with A proxy for Distirubted DB

研究代表者
松尾　啓志（Matsuo, Hiroshi）

名古屋工業大学・工学（系）研究科（研究院）・教授

研究者番号：００２１９３９６

交付決定額（研究期間全体）：（直接経費）　　3,500,000円

研究成果の概要（和文）：本研究では，プロキシによる透過的なトランザクション処理を実現した．D-KVS はデータの強い一貫性を保証しない結果整合性を用いるため，実用的なトランザクション処理機能を持たない．そこで，D-KVS のように RDB相当のトランザクション処理機能を持たない分散 DB にプロキシを導入することでその機能を実現した．また，分散 DB でトランザクション処理を行う場合，高速なトランザクション処理を実現することが困難であった．そこで，プロキシ上でトランザクション処理を行い，分散 DB におけるトランザクション処理のオーバヘッドを削減することで，処理の高速化を行った．

研究成果の学術的意義や社会的意義
従来、安価なシステム上で実用的なトランザクション機能が提供されていなかった分散DBに対して、現在主流となっておりメニーコアとDPDKを用いることにより、安価で高速なトランザクション機能を追加するシステムについて提案を行った。
その結果、現在オープンソースとして公開されているyogabyteDBに比べて、約１０倍高速なトランザクション性能を実現した。

研究成果の概要（英文）：D-KVS does not have a practical transaction processing function because it uses result consistency that does not guarantee strong data consistency. In order to achieve this, we introduced a proxy into the distributed DB that does not have transaction processing functions equivalent to RDB like D-KVS. In addition, it is difficult to achieve high-speed transaction processing when transaction processing is performed on distributed DB. Therefore, we performed transaction processing on the proxy and reduced the overhead of transaction processing in the distributed DB to speed up the processing.

研究分野：分散コンピューティング

キーワード：分散DB　分散トランザクション　メニーコア　DPDK

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

従来，リレーショナルデータベース（RDB）は金融システムや業務システムなど，様々な場面で利用されてきた．RDB では，データはテーブルと呼ばれる二次元の表形式で管理され，リレーションに対して関係代数を用いることで，柔軟なクエリ要求を行うことが可能である．また，ACID　準拠なトランザクション処理をサーポートしており，複数のトランザクションを同時に実行した場合でも，それぞれのトランザクションが逐次的に実行した場合と同一の結果になることが保証される．

SNS, IoT などの普及により，ビッグデータ処理では大量のユーザから生成される大規模なデータの高速な保存･検索処理を行う必要性が高まっている．また，これらのデータは常に生成され，データ量は増加していく一方であり，一台の計算機で全てのデータの入出力やクエリ処理を行うには限界があるため複数の DB ノードを用いてスケールアウトさせる必要がある．しかし，RDB では，複雑な問い合わせ処理や一貫性の保証を行う必要があるため，スケールアウトが困難である．そのため，スケールアウトに適した問い合わせ方法やデータ構造を持つ，分散キーバリューストアが利用されつつある．

しかし、分散キーバリューストアでは、トランザクション機能を提供していないか、もしくは提供していても、簡易な機能しか提供していない場合がほとんどである。

そこで **NewSQL** が開発・運用されている。**NewSQL** は全てのワークロードを実現する単一 **DB** であり，前述したようなマルチデータストアにおける開発複雑性を除去する．**NewSQL** は **Paxos** や **Raft** などの分散合意プロトコルを使用することで，**RDB** のようなデータの強い一貫性を保証しつつ **D-KVS** のようなスケーラビリティを実現する．また，タイムスタンプを使用することで分散環境下でも **RDB** のような相当の **ACID** トランザクション処理機能を提供する．しかし，**NewSQL** ではトランザクション処理を適用する際，ノード間の時刻のズレ以上の時間を待つことでトランザクションを適切な順序で実行するため，その性能を向上するにはより正確な時刻を必要とする．代表的な **NewSQL** である **Google Spanner** は原子時計を使用することで正確なタイムスタンプを取得し，高いトランザクション処理性能を実現する．しかし，原子時計などの特別なハードウェアの導入は運用が複雑であり，システムの大幅な変更が必要となる．そこで，**Spanner** のクローンである **CockroachDB** や **YugabyteDB** は原子時計などの特別なハードウェアを使用せず **RDB** 相当のトランザクション処理機能を実現する．**NTP** などの数十ミリ秒程度の同期精度を持つクロック同期サービスを利用することでトランザクションの順序制御を行う．しかし，原子時計から得られるタイムスタンプと比較して時間のズレが大きいため，トランザクション処理における実行の待ち時間が増加し **Spanner** よりトランザクション処理性能が大幅に低下する．

２．研究の目的

本研究では集中型のプロキシを分散 **DB** に導入し，プロキシ上でトランザクション処理を行うことで，その機能を持たない **D-KVS** に対しその機能を提供しつつ，**NewSQL** に匹敵するトランザクション処理性能を向上することを目的とする。

３．研究の方法

本研究ではメニーコアを活用した高性能なプロキシによる透過的なクエリ処理手法を提案する．プロキシを導入することで，RDB 相当の ACID 準拠なトランザクション処理機能を持たない分散 DB にその機能を提供する．さらに，プロキシ上でトランザクション処理を行うことで，分散 DB におけるトランザクション処理のオーバヘッドを削減し，トランザクション処理の高速化を実現する．

3.1 要件

本提案手法が満たすべき要件として以下の３つがある．

(a) トランザクション処理の性能向上
(b) RDB 相当の ACID 準拠なトランザクション処理
(c) 透過性の実現および複雑性の除去

１つ目の要件として，トランザクション処理の性能向上がある．従来，分散 DB でデータの一貫性保証のために排他制御を行う場合，分散合意プロトコルを使用する必要があった．
しかし，分散合意プロトコルを使用する場合，ノード間での通信が多数必要となるため効率的ではない．そこで，プロキシ上で排他制御を行うことで，分散合意プロトコルを除去し，分散 DB におけるトランザクション処理の高速化を実現する．

２つ目の要件として，RDB 相当の ACID 準拠なトランザクション処理がある．D-KVS などの分散 DB は RDB 相当のトランザクション処理機能を持たない．したがって，金融システムや業務システムなどトランザクション処理機能を必要とするようなシステムに D-KVS を導入することは困難である．そこで，プロキシ上でトランザクション処理を行うことで，RDB 相当の ACID 準拠なトランザクション処理機能を持たない分散 DB にその機能を提供する．

３ つ目の要件として，透過性の実現および複雑性の除去がある．プロキシがバックエンドの分散 DB のプロトコルに対応することで，クライアントはプロキシを意識すること無くクエリを実行可能である．また，プロキシがトランザクション処理機能を提供するため，クライアントは RDB や D-KVS の両方を使用する必要はなく開発や運用の複雑性を除去可能である．さらに，プロキシ

では原子時計などを使用すること無く高速にトランザクション処理を実現可能であるため，特別なハードウェアを導入するなどという複雑性も除去可能である．

3.2 全体構成

提案手法が満たすべき 3 つの要件を達成するために，以下に示す 3 つの処理をプロキシ上で行う．

(1) プロキシによる排他制御
(2) プロキシによるトランザクション処理
(3) プロキシによる透過的なクエリ処理

3.2.1 プロキシによる排他制御

プロキシはロックによる排他制御を行う．プロキシがクライアントからの要求をすべて受け付け，ロック獲得者を決定し排他制御を行うことで，分散合意プロトコルを除去した高速なトランザクション処理を実現する(図１)．さらに，NewSQL で見られた Commit Wait によるトランザクション処理適用時に発生する待ち時間を排除することも可能となる．

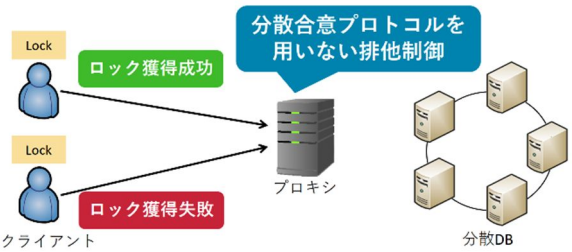

図 1　プロキシによる排他制御

3.2.2 プロキシによるトランザクション処理

プロキシ上でトランザクション処理を行うことで，RDB 相当の ACID 準拠なトランザクション処理機能を持たない分散 DB にその機能を提供する(図 2)．プロキシが提供するトランザクション処理は ACID 特性を満たす必要がある．そこで，それらの特性を満たすために新たなクエリの実装や排他制御，分散 DB による処理結果の永続化を行う．

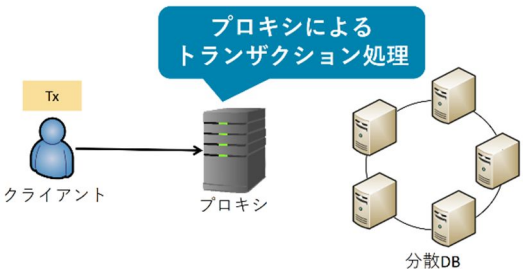

図 2　プロキシによるトランザクション処理

3.2.3 プロキシによる透過的なクエリ処理

RDB や D-KVS を組み合わせて使用するマルチデータストアではアプリケーションの開発や各 DB の運用・管理が複雑になるという問題があった．また，NewSQL である Spanner ではトランザクション処理高速化のために原子時計などの特別なハードウェアを用いるという複雑性があった．そこで，それらの複雑性を除去するためにプロキシによる透過的なクエリ処理が必要となる．分散 DB に提案手法を適用することで，クライアントは RDB や D-KVS を組み合わせて使用する必要がないため，マルチデータストアに見られた複雑性を除去可能である．さらに，プロキシによる透過的なトランザクション処理により特別なハードウェアを用いるという複雑性を除去する．

3.2.4 トランザクション処理の性能向上

プロキシ上で排他制御を行うことで分散 DB における排他制御のオーバヘッドを除去する．まず，プロキシによる排他制御により分散合意プロトコルを除去する(図 3)．従来，分散 DB で排他制御を行う場合，DB ノード間で分散合意プロトコルを使用する必要があるため，排他制御に必要な通信コストが大きいという問題があった．
しかし，プロキシ上で排他制御を行い，ロック獲得者を決定することで，プロキシが一貫した結果を返すことが可能であるため，DB ノード間における一貫性制御のための通信コストを削減することが可能である．さらに，NewSQL でトランザクション処理を行う場合，Commit Wait によりトランザクション処理適用時における待ち時間が存在した．しかし，プロキシによる排他制御によりそのオーバヘッドも除去可能である．

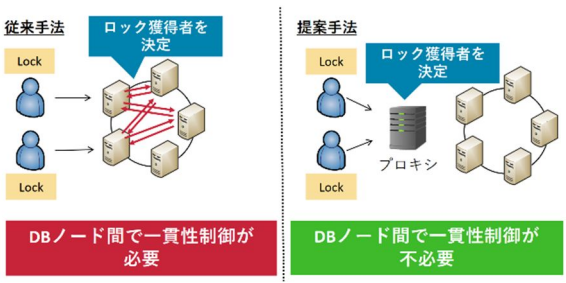

図 3　分散合意プロトコルの除去

3.2.5 RDB 相当の ACID 準拠なトランザクション処理

プロキシが提供するトランザクション処理は ACID 特性を満たす必要がある．まず，Atomicity を満たすためにトランザクションの開始，適用，破棄処理が必要となる．さらに，Consistency と Isolation を満たすためにロックによる排他制御が必要となる．最後に，Durability を満たすためにトランザクション処理結果の永続化が必要となる．以下では，これらの特性を達成する

ために必要となる機能の詳細を述べる．

(a) Atomicity の達成
Atomicity を達成するために，以下に示すトランザクションを制御するためのクエリを新たに 3 種類実装した．

・BEGIN トランザクション処理の開始
・COMMIT トランザクション処理の適用
・ROLLBACK トランザクション処理の破棄

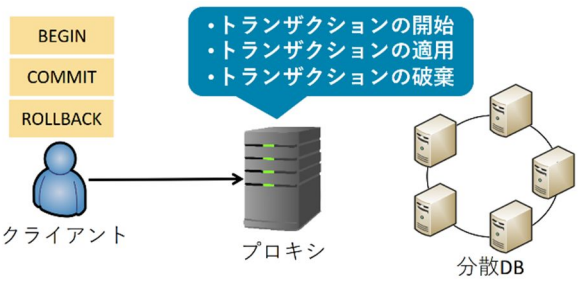

図 4 Atomicity の実現

クライアントはそれら 3 種類のトランザクション制御クエリを使用することでトランザクション処理を実行可能となる(図４)．Consistency や Isolation を達成するために，プロキシ上でロックによる排他制御を行う．プロキシ上でロックの管理を行うことで分散合意プロトコルを用いない排他制御を実現する．それにより，プロキシが一貫した結果をクライアントへ返すことが可能であるため Consistency を達成する．さらに，ロックにより各トランザクション同士で影響を与え合うことがないため Isolation も達成する．

(b) Consistency・Isolation の達成
Consistency や Isolation を達成するために，プロキシ上でロックによる排他制御を行う．プロキシ上でロックの管理を行うことで分散合意プロトコルを用いない排他制御を実現する．それにより，プロキシが一貫した結果をクライアントへ返すことが可能であるため Consistency を達成する．さらに，ロックにより各トランザクション同士で影響を与え合うことがないため Isolation も達成する．

(c) Durability の達成
Durability を達成するために，分散 DB 上でトランザクション処理結果の永続化を行う．プロキシは COMMIT クエリによりトランザクションを適用する際，トランザクション処理における書き込みクエリを分散 DB に送信する．そして，分散 DB でその書き込みクエリを実行することでトランザクション処理結果の永続化を行う．

3.2.6 透過性の実現および複雑性の除去
プロキシがバックエンドの分散 DB のプロトコルに対応することで，クライアントはプロキシを意識すること無くクエリを実行可能である．また，プロキシがトランザクション処理機能を提供するため，クライアントは RDB や D-KVS の両方を使用する必要はなく開発や運用の複雑性を除去可能である．さらに，プロキシでは原子時計などを使用すること無く高速にトランザクション処理を実現可能であるため，特別なハードウェアを導入するなどという複雑性も除去可能である．

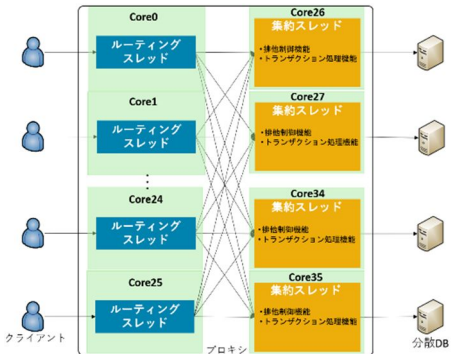

図 5　メニーコアを活用したクエリ処理

3.3 メニーコアを用いた実装方式
分散 DB にプロキシを導入する場合，プロキシがクライアントからのクエリを全て受け付けるため，高いクエリ処理能力が必要となる．そこでプロキシはメニーコアを活用し複数のスレッドでクエリ処理を行う．プロキシ内ではルーティングスレッドと集約スレッドの 2 種類が動作している．ルーティングスレッドはクライアントからクエリを受信し，クエリの Key から宛先を計算してルーティングを行う．集約スレッドはクエリの集約を行い，それを分散 DB へ送信する．プロキシにおける排他制御機能やトランザクション処理機能は集約スレッド上に実装した(図５)．各ルーティングスレッドがクライアントからのクエリを受信するため，Key が同一のクエリを処理する場合がある．そこで，ルーティングスレッド上に排他制御機能を実装する場合，ロック管理テーブルの一貫性制御のためにルーティングスレッド同士でスレッド間通信を必要とするため，プロキシの性能悪化が考えられた．ここで，ロック管理テーブルをルーティングスレッド間で共有する実装方式が考えられるが，ロック管理テーブルの排他制御を必要とするため，これもプロキシの性能悪化が考えられた．一方で，集約スレッドが処理するクエリはルーティングスレッドによりルーティングされたものであるため，Key が同一のクエリは同一の集約スレッドが処理する．そのため，各集約スレッドがロック管理テーブルを保持することで，スレッド間通信やロック管理テーブルの排他制御を必要としない実装が可能で

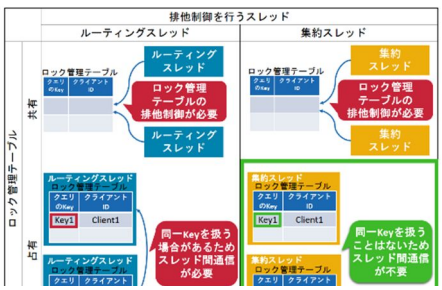

図 6　各スレッドによる排他制御・トランザクション実装

ある．
したがって，プロキシにおける排他制御機能やトランザクション処理機能は各集約スレッド上に実装した(図６)．

４．研究成果

以下に示す 2 種類の評価結果を分析して提案手法の有効性を確認する．

(a)トランザクション処理機能を実装したことによるオーバヘッドの評価:プロキシによるトランザクション処理のオーバヘッドが小さいことを確認するために，トランザクション処理機能を実装前のプロキシと実装後のプロキシの性能比較を行った．

(b) プロキシによるトランザクション処理の性能評価:プロキシによるトランザクション処理の有効性を確認するために，NewSQL (YugabyteDB)とのトランザクション処理性能の比較を行った．評価に用いた物理サーバの仕様をそれぞれ表 1，表 2 に示す．

	プロキシ
OS	Ubuntu 18.04 LTS
CPU	Intel Core i9-7980XE 2.60 GHz (18 cores/36 threads)
Memory	64GB
NIC	Intel X550T (10GbE, 2ports)

表 1 プロキシに用いる計算機

	クライアント	DB nodes
OS	Ubuntu 18.04 LTS	Ubuntu 18.04 LTS
CPU	AMD Ryzen Threadripper 2990WX 3.0 GHz (32 cores/64 threads)	Intel Core i5-4460 3.20 GHz
Memory	64GB	16GB
NIC	Intel X540-AT2 (10GbE)	

表 2 クライアント、DB ノードに用いる計算機

4.1 トランザクション処理機能を実装したことによるオーバヘッドの評価

提案手法をプロキシに実装したことによるオーバヘッドが小さいことを確認するために，トランザクション処理機能を実装前のプロキシと実装後のプロキシの性能比較を行った(図 7). 評価ではクライアントに YCSB を分散 DB に Cassandra を 10 ノード使用した．クライアントの性能を飽和させるため，YCSB のスレッド数を 160 として評価を行った．YCSB には Workload A から F までの 6 種類のワークロードが存在するが，プロキシが対応していない Workload E を除いた 5 つのワークロードを用いて評価を行った．評価結果を図に示す．グラフの縦軸はスループット，横軸は各ワークロードを表している．また，青色の棒グラフがトランザクション処理機能を持たないプロキシ，黄色の棒グラフがトランザクション処理機能を実装したプロキシの評価を表している．Workload A では 1.2%，Workload B では 0.2%，Workload C では 1.1%，Workload D では 1.3%，Workload F では 8.8%スループットが低下する結果となった．これは，トランザクション処理機能を実装したプロキシによるロック状態やトランザクション状態の確認などのオーバヘッドである．しかし，この結果からトランザクション処理機能を実装したことによるオーバヘッドは小さく許容可能である．

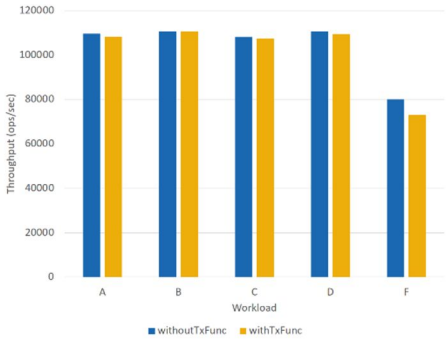

図 7 オーバヘッドの評価

4.2 プロキシによるトランザクション処理の性能評価

トランザクション処理における提案手法の有効性を確認するために，NewSQL である YugabyteDB とのトランザクション処理の性能比較を行った(図 8). 評価では，クライアントとして 10 回の書き込み処理を行うトランザクションを 1000 回実行するマイクロベンチマークを，分散 DB には YugabyteDB を 10 ノード使用した．評価結果を図に示す．

グラフはヒストグラムであり，縦軸は頻度，横軸はトランザクション処理の実行時間を表す．

青色の実線は YugabyteDB によるトランザクション処理性能，黄色の実線は提案手法によるトランザクション処理性能を表す．ヒストグラムから，提案手法によるトランザクション処理は YugabyteDB のそれと比較して，全体として短い実行時間で処理できている．これは，提案手法により，分散合意プロトコルを除去しその通信コストを削減したことや，NewSQL に見られた Commit Wait によるトランザクション処理適用時における待ち時間を除去したことに起因する．さらに，評価結果の平均値，中央値，最大値，最小値を表３に示す．中央値を比較すると提案手法によるトランザクション処理は YugabyteDB のそれと比較して約 6.4 倍高速に動作することを確認した．

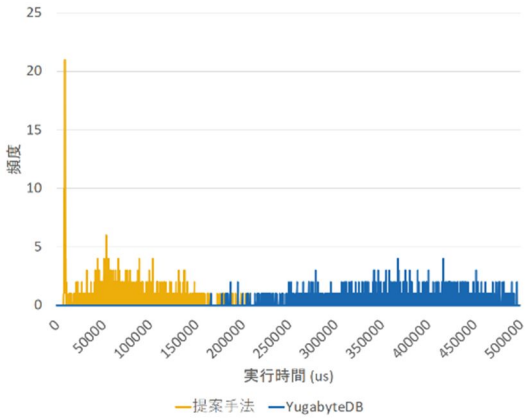

図 8 トランザクション処理性能評価

実行時間 (us)	平均値	中央値	最大値	最小値
提案手法	65630	59045	1340799	7533
YugabyteDB	383857	378630	817382	166552

表 3 トランザクション処理の実行時間

５．主な発表論文等

〔雑誌論文〕　計8件（うち査読付論文　5件／うち国際共著　0件／うちオープンアクセス　1件）

１．著者名 宮本 基志，三輪 竜也，川島 龍太，松尾 啓志	４．巻 vol.2020-OS-150
２．論文標題 分散KVSにおけるプロキシを用いたトランザクション実装	５．発行年 2020年
３．雑誌名 情報処理学会技術報告オペレーティングシステム	６．最初と最後の頁 1-7
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Ryuya Miwa, Motoshi Miyamoto, Ryota Kawashima, Hiroshi Matsuo	４．巻 1
２．論文標題 Transparent Transaction Processing with A High-Performance Proxy for Distributed KVS	５．発行年 2020年
３．雑誌名 The Eighth International Symposium on COmputing and Networking Workshops (CANDARW)	６．最初と最後の頁 0-0
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/CANDARW51189.2020.00051	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 早瀬 大智，稲垣 英夫，川島 龍太，松尾 啓志	４．巻 2019-OS-147(6)
２．論文標題 Apache Sparkにおけるブルームフィルタを用いたSQL処理の高速化手法の提案	５．発行年 2019年
３．雑誌名 情報処理学会技術報告オペレーティングシステム	６．最初と最後の頁 1-6
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 三輪 竜也，川浪 大知，川島 龍太，松尾 啓志	４．巻 2019-OS-147(3)
２．論文標題 分散KVSにおけるメニーコアを活用したプロキシによるクエリ集約及び排他制御	５．発行年 2019年
３．雑誌名 情報処理学会技術報告オペレーティングシステム	６．最初と最後の頁 1-6
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Yuki Taguchi, Ryota Kawashima, Hiroki Nakayama, Tsunemasa Hayashi, Hiroshi Matsuo	4．巻 E102-D
2．論文標題 Fast Datapath Processing based on Hop-by-Hop Packet Aggregation for Service Function Chaining	5．発行年 2019年
3．雑誌名 IEICE Transactions on Information and Systems	6．最初と最後の頁 2184-2194
掲載論文のDOI（デジタルオブジェクト識別子） 10.1587/transinf.2018EDP7444	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

1．著者名 Daichi Kawanami, Masanari Kamoshita, Ryota Kawashima, Hiroshi Matsuo	4．巻 1
2．論文標題 A Proxy-based Query Aggregation Method for Distributed Key-Value Stores	5．発行年 2018年
3．雑誌名 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)	6．最初と最後の頁 pp.78-83
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/W-FiCloud.2018.00018	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Hideo Inagaki, Tomoyuki Fujii, Ryota Kawashima, Hiroshi Matsuo	4．巻 1
2．論文標題 Adaptive Control of Apache Spark's Data Caching Mechanism Based on Workload Characteristics	5．発行年 2018年
3．雑誌名 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)	6．最初と最後の頁 pp.64-69
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/W-FiCloud.2018.00016	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Hideo Inagaki, Tomoyuki Fujii, Ryota Kawashima, Hiroshi Matsuo	4．巻 1
2．論文標題 Improving Apache Spark's Cache Mechanism with LRC-based Method using Bloom Filter	5．発行年 2018年
3．雑誌名 The Sixth International Symposium on Computing and Networking Workshops (CANDARW)	6．最初と最後の頁 pp.496-500
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/CANDARW.2018.00096	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計0件

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究分担者	川島　龍太 (Kawashima Ryota) (00710328)	名古屋工業大学・工学（系）研究科（研究院）・准教授 (13903)	

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関