6W-2

大規模センサネットワーク構築のためのシームレスなデータ共有

岩田寛生† 末松直樹†† 宮崎敏明†

会津大学大学院コンピュータ理工学研究科† 会津大学コンピュータ理工学部††

1. はじめに

近年、環境情報の収集にセンサネットワーク(SN)が多用されている。各 SN で取得したセンサデータの管理は一般にローカルで行われており、複数 SN 間のデータ共有は環境の違いなどからあまり実施されていない。しかし、地理的に離れた場所同士でセンサデータ共有ができれば、ユーザは地理的、時間的に大規模なセンサデータを扱う事が可能となり実用的に大きな意味を持つ。上記を実現するには、柔軟に個々のセンシング環境に適応しつつ、取得したセンサデータを従来通りローカルに管理できる一方、各ローカルセンサデータを統合して大規模データベース(DB)が構築できることが望ましい。

本稿では、ピア・ツー・ピア(P2P)ベースの分散 DB を用いることにより、各 SN で取得したセンサデータはローカルで管理しつつ、それらが複数統合された大規模センサネットワークにおいては、ローカルに管理しているデータに手を加えることなくシームレスに大規模センサデータを管理可能とする手法を提案する。

2. 大規模データの共有と管理

環境情報収集のために設置された小規模 SN をインターネットで接続して構成された全国を網羅する大規模 SN を想定すると、膨大なセンサデータを管理する必要がある。さらに、各 SN はセンサの種類や個数などが異なるため、データ管理機構には、それら差異も許容する柔軟性が求められる。先行研究として、異種 SN 環境を想定したセンサデータの位置情報による統合手法[1]や異種センサデータを複数のピア(以下、単にピア)で管理、検索するために PostgreSQL を改良した分散 DB TomuDB[2]などがある。TomuDB はセンシング領域や粒度情報に基づいた効率的な検索が可能であるが、ピア同士のコネクションがツリー構造のため単一障害点が存在するという問題がある。また、既存技術は、リレーショナル DB(RDB)を用いたものが主である。しかし、RDB は事前に決めたスキーマに従ってデータを保存する必要があり、新規センサの追加など、保存すべきデータが動的に変化する SN のデータ管理には適さない。一方、NoSQL と呼ばれる分散 DB、例えば Hadoop HBase[3]、Cassandra[4]などが、ビッグデータの管理手法として近年注目されている。HBase はマスタ/スレーブモデルを採用しており、単一障害点が存在するため、本稿で目指すピアも含めた柔軟な管理という点において問題が残る。一方、Cassandra は P2P モデルを採用しており、複数のピア同士でリング状のクラスタを構成するため単一障害点がない。また、Cassandra は列指向 DB であり、動的にカラムを追加してデータを管理するため SN におけるデータの動的な変化には、カラムを追加するのみで良く、柔軟に対応することができる。本稿では大規模 SN 構築の際に求められる、定期的な書き込み要求や地理的に離れた環境の異なる SN 間のデータ共有、SN 管理者によるデータ管理などの問題に対処するために、Cassandra をベースとして、データ保存場所に焦点を当てた解決方法を提案する。

Seamless Data-sharing for Constructing Large-scale Sensor Network

†Hiroki Iwata, ††Naoki Suematsu, †Toshiaki Miyazaki

†Graduate School of Computer Science and Engineering, The University of Aizu

††School of Computer Science and Engineering, The University of Aizu

3. センサデータ共有の最適化

2 章で述べた Cassandra のデータ保存方法として、分散保存と非分散保存時の SN からの書き込み(Write: 実線)、Client のデータ読み込み(Read: 点線)のデータの流れを図 1 に示す。図上部の分散保存では保存するデータのキー毎にハッシュ計算(ハッシング)することで、クラスタ内の保存先ピアが決定される。また、Client からの Read 要求もハッシングを使いデータを保有するピアへと送られる。一方、図 1 下部の非分散保存では、キーにより目的のピアにデータを直接書き込むことが可能である。Client もデータのキーが既知である場合、直接データを保持するピアを指定した Read 要求を送ることができる。また、非分散保存と同様にクラスタ内の他ピアからのデータも取得可能である。

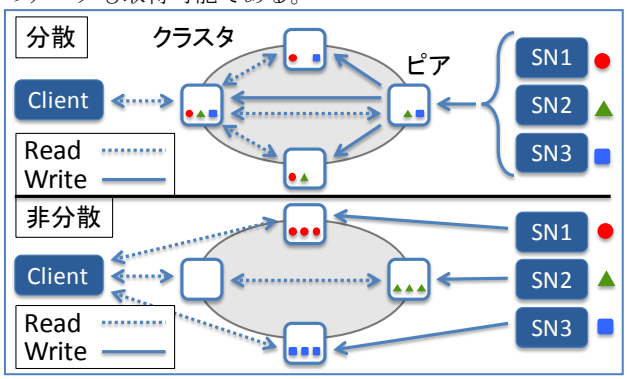

図 1 Cassandra における分散・非分散保存の概要

3.1 ローカル内で完結するデータ書き込み

大規模 SN を構築するためには、複数の SN が地理的に離れているため通信帯域・遅延の影響を受けること、及び、定期的にセンサデータの書き込みが発生することを考慮しなければならない。分散保存ではデータをクラスタ全体に保存するため、Read/Write 共にピア間の通信が必要になり、必然的に通信による影響を受ける。一方で非分散保存では、SN 内で書き込みが完結するだけでなくクライアントからの読み込みも①キー指定での検索、②クラスタ全体の検索、が可能である。さらに、非分散保存で管理することで、各 SN の管理者も他 SN のことを考えることなく、シームレスなデータ共有が可能である。

3.2 位置情報と取得時間を用いた検索範囲の絞込み

センサデータは特性として位置情報、取得時間、種類情報というメタデータを持っており、分散・非分散保存に関わらず、これら要素を指定することでデータ検索を行う。また、非分散保存においては 3.1 章で述べた様にキー指定による検索範囲の絞込みが可能であるため、データを保存する際にキーとして位置と時間を組み合わせたものを用いることで目的のピアのみに要求を出すことができる。テーブル生成のためのクエリは図 2 に示した通りである。PRIMARY KEY として location_time(位置情報と取得時間), sensor_property(センサの種類),

sensor_value(センサの値), sensor_id(センサの ID)を指定している。これは SQL に似た CQL という Cassandra を RDB 風に使用するときの記法であり、実際の Key-Value 構造を持つ Cassandra 上では位置と時間がキーとなる。また、location_time 以外の PRIMARY KEY にもインデックスの追加設定を行うことで高速な検索が可能となる。

```
CREATE TABLE db.Sensors (
    location_time       text,
    sensor_property     text,
    sensor_value        double,
    sensor_id           text,
    lat                 double,
    lng                 double,
    PRIMARY KEY (
        location_time,
        sensor_property,
        sensor_value,
        sensor_id
    ));
```

図 2 Cassandra でのテーブル構造記述クエリ

4. 分散・非分散保存の性能比較実験

4.1 実験環境

表 1 の"共通ソフトウェア環境"に記載した内容で全ての PC を統一した。また、ハードウェア環境は表の構成の PC を 5 台 Cassandra ピア用として用意しクラスタを構成した。さらに、読み込み用クライアントとして表 1 下部に示す PC を別途 1 台用意した。

表 1 実験環境

共通ソフトウェア環境	
OS	CentOS release 6.4 (Final) x86_64
JDK	OpenJDK 1.7.0_45
Cassandra	Server: 2.0.x Client: Cassandra-driver-core 1.0.3
ハードウェア環境（Cassandra ピア）	
CPU	Intel® Core™ i7-3770 CPU @ 3.40GHz
メモリ	8 GB
ディスク	サイズ: 1 TB 回転速度: 7200 RPM
ハードウェア環境（読み込み用クライアント）	
CPU	Intel® Core™ i5-2400 CPU @ 3.10GHz
メモリ	6 GB
ディスク	サイズ: 500 GB 回転速度: 7200 RPM

4.2 実験

大規模 SN では SN 毎に個別に書き込み要求が発生することを想定し、複数台同時書き込みを行う。データ格納のための Cassandra のピア 5 台を書き込み要求クライアントとしても兼用し同時に書き込みを行う。読み込み要求に関しては、上記書き込みが終わった後に 1 台のクライアントから読み込み要求を行うものとした。表 2 に具体的に書きこみ/読み込みに用いる CQL 文を示す。なお、表中で{}で囲まれたものに関しては固定長で適宜変更して書き込みを行う。

(ア) 複数 SN による同時書き込み

複数同時書き込みのため、表 2 の属性が“書き込み”のクエリ要求 100,000 件を 1 分毎に発行するクライアントをピア毎に用意し同時に実行する試行を 600 回行う。その結果、総データ量は 300,000,000 レコード(100,000 件×600 回×5 台)となり、表 2 の読み込みの際のクエリの対象となるデータは 1 ピアあたり 6,000,000 件とした。

(イ) データ読み込み

一度の要求で取得するデータ件数を 100～100,000 件とする。表 2 の属性“読み込み/キー指定”と“読み込み/キー指定無し”のクエリをそれぞれ読み込みクライアントから発行する。また、分散保存の場合にはキーのハッシュ値を用いるためキー指定検索を行うことはできない。

表 2 書きこみ/読み込みに使用する CQL 文

属性	CQL文
書き込み	*INSERT INTO db.Sensors (location_time,sensor_property,sensor_value, sensor_id,lat,lng) Values('{location}:{time}','TEMP',{value},'NODE{id}',37.4539,140.2825);*
読み込み/キー指定	*SELECT * FROM db.Sensors WHERE location_time >= '{location0}:0' AND location_time < '{location1}:0 AND sensor_property = 'TEMP' AND sensor_value = {value} limit {limit};*
読み込み/キー指定無し	*SELECT * FROM db.Sensors WHERE sensor_property = 'TEMP' AND sensor_value = {value} limit {limit};*

4.3 実験結果

表 3 に書き込み実験結果を示す。非分散保存することで複数台同時書き込み速度では分散保存の 1.52 倍となった。また、表 4 に読み込み実験結果を示す。読み込み速度では全体を通して非分散の試行が速いという結果であった。非分散保存でピア指定した結果、1,000~100,000 件の読み込み速度で最速の結果になり、書き込み、読み込み両面において分散保存よりも良い性能を示した。

表 3 複数 SN による同時書き込み実験結果

書き込み速度		
	平均(ms)	速度比(倍)
分散	(*1) 8,026.34	1.00
非分散	**5,285.92**	**1.52**

*1 分散の場合の書き込み要求のタイムアウトによる失敗を除く。

表 4 1 クライアントのデータ読み込み実験結果

読み込み速度			
	分散(ms)	非分散(ms)	
	キー指定無し	キー指定	キー指定無し
100	1,255	555	**256**
1,000	2,486	**1,305**	1,762
10,000	18,059	**10,558**	13,376
100,000	284,377	**94,859**	127,807

5. 結論

本稿では、P2P ベースの分散 DB である Casandra を用いて、個別のセンサネットワーク(SN)で取得したセンサデータをローカルで管理し、それらを統合した大規模 SN において、ローカルに管理しているデータに手を加えることなくシームレスに大規模センサデータを管理可能とする手法を提案した。提案手法は、複数台でデータを分散保存せず、SN 毎にローカルで管理することで、Read/Write 性能を犠牲にすることなく、他の SN を考える必要のないシームレスなデータ管理が可能であることを、実験により示した。今後は、全国規模のネットワークを用いて、通信遅延や帯域制限など影響を詳細に調べて行く。

謝辞

本研究の一部は，総務省戦略的情報通信研究開発推進制度（SCOPE No.121802001）および科研費(No.23500095)の支援を受けて実施したものである。

参考文献

[1] 白石陽, 安西祐一郎, “センサデータの視覚化のためのインクリメンタルな空間集約手法,” 情報処理学会論文誌：データベース, Vol.45, No.7, pp.63-76, 2004
[2] 岩井将行, THEPVILOJANAPONG Niwat, 石塚宏紀, 中村陽一, 金井圭介, 白石陽, 戸辺義人, “TomuDB : 都市空間センサノード情報を扱うデータベースシステム,” 電子情報通信学会技術研究報告. USN, ユビキタス・センサネットワーク, Vol.108, No.138, pp.13-18, 2008
[3] HBase http://hbase.apache.org/
[4] Cassandra http://cassandra.apache.org/