Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

センサデータ収集システムにおける分散データベースの性能評価

松浦伸彦[†1] 鈴木誠二[†2] 峰野博史[†2]
太田賢[†3] 水野忠則[†4]

近年，センサ技術や携帯端末の高性能化に伴い，サーバに蓄積されたデータを利用した携帯端末向けサービスが続々と登場している．センサは発生するデータをサーバに書き込み，サービスはサーバからデータを読み込みユーザに向けて情報を提供する．しかし，このようなサービスでは，データベースに対しての読み書きのリクエスト数や蓄積されているデータ量が膨大になる．これにより，高頻度な書き込みと読み込みに対応できるだけの性能，ならびスケーラビリティがデータベースに対して必要とされる．そこで本研究では，データストリーム管理システムと分散データベース管理システムを組み合わせ，高頻度な読み書き両方に対応し，高いスケーラビリティを持ったセンサデータ収集システムの提案・評価を行う．高齢者見守りサービスのような高負荷なサービスを想定したとしても，書き込みに最適化された分散データベース管理システムの利用と，高いスケーラビリティを利用して複数台でクラスタを形成することで十分対応可能であることを確認した．また，高頻度な読み込みをデータストリーム管理システムに任せることでスループット減少を抑え，バルクインサートの利用で大幅な性能向上が期待できることを確認した．

Performance Evaluation of Distributed Database in Sensor Data Collection System

NOBUHIKO MATSUURA,[†1] SEIJI SUZUKI,[†2]
HIROSHI MINENO,[†2] KEN OHTA[†3]
and TADANORI MIZUNO[†4]

In recent year, a life support service using sensor and mobile phone dates has attracted attention, along with the development of sensor and mobile teminal. The sensor write data on the server, and the service read data from the server to provide information to user. However, a system read or write large amount of data to a database of server. Thus, the database requires a high read and write performance and scalability. To solve the problem, we propose and evaluate the sensor data collection system with distributed database management system and data stream management system. This system has high read and write performance and scalability. We confirm that the system can handle heavy workload service, such as a watching senior citizen service, using cluster of the write optimized and scalable database.

1. はじめに

近年，センサデータや携帯端末の位置情報などのデータを利用した携帯端末向けサービスが続々と登場している．その一例として，タクシーのプローブデータによる交通渋滞情報提供[1)]，電子機器の制御による省エネサポート[2)]，高齢者の見守り支援[3)] など，センサから収集したデータを利用した生活の質を高めるサービスを実現できる環境が整いつつある．このようなサービスを考えた際に，一般的にクライアント，サーバ，データベースという三層構造で構築される事が多い．センサで収集されたデータはデータベースに保存され，データベースに保存されているデータを用いてサーバはクライアントへサービスを提供する．データの蓄積や処理はサーバで全て行われるため，携帯端末等のモバイル機器から時間や場所に捕らわれることなくサービスを受けることができ，またユーザ間でのデータ共有も容易となる．

センサデータを用いた一般向けサービスを提供する場合，データベースには大量のリクエストが送られ，膨大な負荷が発生することがある．例えば高齢者見守り支援を考えた時，近年では 1 人暮らしの高齢者が約 387 万人居る[4)] ことを踏まえ，各世帯にセンサを配置して 1 分毎にデータを収集したとすると，1 秒間に 64,500 件もの書き込みリクエストがサーバに到達することとなる．そのため，センサデータ収集システムにはこの膨大な書き込みに対応することのできる性能とスケーラビリティが求められる．また，読み込みリクエストに関しても，モニタリング処理等で相当な負荷が予想されるために対応が必要となる．

そこで本研究では，データの読み書き両方の処理を効率良く処理することのできるセンサデータ収集システムを検討し，性能評価ならびスケーラビリティに対して検証を行う．

†1 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
†2 静岡大学情報学部
Faculty of Informatics, Shizuoka University
†3 株式会社 NTT ドコモ 先進技術研究所
NTT DOCOMO, Inc. Research Laboratories
†4 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

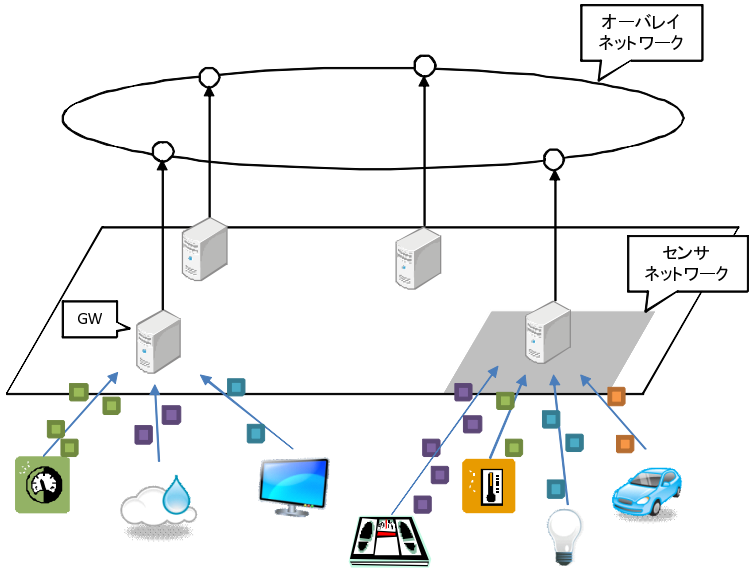


図 1 センサデータ収集システムの分散アーキテクチャ

2. 関 連 研 究

2.1 P2P ネットワーク

Peer-to-Peer(P2P) ネットワークは，従来のネットワーク上に仮想的な P2P 用のネットワーク (オーバレイネットワーク) を構築し，複数のノードを相互接続させて直接通信を行う．そして，このオーバレイネットワーク上に分散ハッシュテーブル (DHT) を形成し，データの分散管理も行う．DHT は Key と Value の組を分散管理する手法であり，Key のハッシュ値などを利用して適当なノードを保存先として選択する．保存したデータを取得したい場合は，Key からハッシュ値を求めて保存先を求め，対応する Value を取り出す．DHT の構築には，Chord[5]，Tapestry[6]，Kademlia[7]，Skipgraph[8] 等のアルゴリズムが利用され，データの読み書きの性能は各アルゴリズムに依存する．

P2P ネットワークを用いることで，**図 1** のようなセンサデータ収集システムを構築することができる．センサネットワーク毎にゲートウェイ (GW) と呼ばれる外部へのデータ転送を制御するノードを配置し，各 GW を P2P ネットワークで相互接続させ，データの処理を行う．このような構成にすることで負荷を各 GW に分散させることができ，もし全ての GW の負荷が高くなったとしても GW を新たに追加することで負荷を分散させることができる．しかし，全体の性能が GW 間のネットワークの遅延や GW の性能に影響され，データの読み書きが安定しないという課題がある．

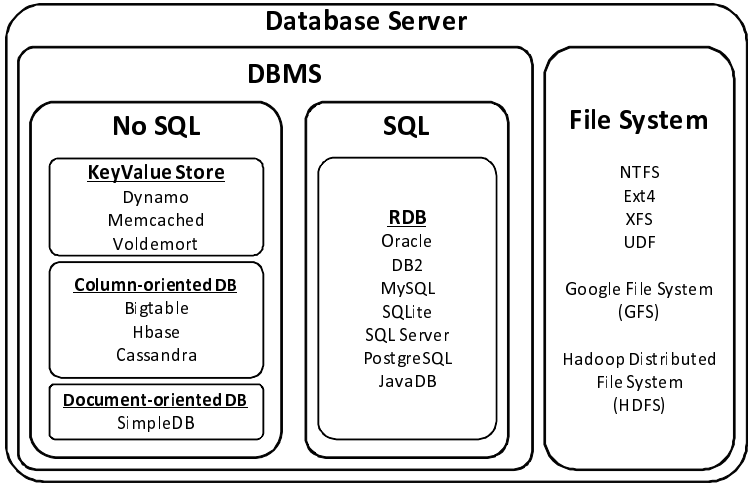


図 2 DBMS の種類

2.2 データベース管理システム

データベース管理システム (Database Management System，DBMS) は，データベースを構築する際に利用されるシステムである．データ形式や利用方法が一般化されており，DBMS を用いる事でアプリケーションからデータ管理を独立させることができる．

図 2 にデータベースサーバの構成を示す．従来，DBMS の構築には SQL を用いるリレーショナルデータベース (Relational Database，RDB) が利用されてきた．しかし，RDB の分散化にはシャーディングなどのアプリケーション側での対応が必要であり，近年では NoSQL と呼ばれるデータベース側で分散機能を持つシステムが登場している．NoSQL は “Not Only SQL” の略であり，SQL を利用しないシステムを指す．代表的な NoSQL を**表 1** にまとめる．データを保存する形式であるデータモデルに関しては，Key-Value Store(KVS)，列指向データベース (Column-oriented Database)，ドキュメント指向データベース (Document-oriented DB) がある．KVS は，キーとバリューの組でデータを保存する．列指向データベースは，KVS

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

表 1 NoSQL の種類

名前	データモデル	CAP 理論	分散方法	永続化方法
Cassandra	Column-oriented	AP	Consistent Hash	Memtable/SSTable
HBase	Column-oriented	CP	Sharding	Memtable/SSTable on HDFS
CouchDB	Document-oriented	AP	Consitent Hash	Append-only B-tree
Riak	Document-oriented	AP	Consistent Hash	?
MongoDB	Document-oriented	CP	Sharding	B-tree
Tokyo Cabinet	Key-value	AP	Consistent Hash	Hash or B-tree
Voldemort	Key-value	AP	Consistent Hash	Pluggable
Redis	Key-value	CP	Consistent Hash	In-memory with background snapshots
Scalaris	Key-value	CP	Consistent Hash	In-memory only

の発展形であり，バリューの部分に複数の値を保存することで擬似的に RDB のような表形式を構成する．ドキュメント指向データベースは，JSON 形式でデータを保存する．一般的に，NoSQL は P2P ネットワークなどを使って複数のサーバを相互接続させており，データを分散管理することでスケーラビリティと可用性を高めている．また，一貫性 (Consistency)，可用性 (Availability)，分散性 (Partition) の中の 3 つを分散システムは同時に満たすことができないという CAP 理論に注目すると，すべての DB が分散データベースの特性として分散性を含んでいることが分かる．

分散データベースの定量的な性能評価手法に関して YCSB[9] という論文が注目されている．YCSB では，DBMS に対して大量の読み書きのリクエストを発生させ，スループットや遅延について定量的な測定を行っている．評価は Cassandra[10]，HBase[11]，PNUTS[12]，MySQL(シャーディング構成) の 4 つを対象に行われ，評価結果により以下のように分類している．

- 読み込み最適化: MySQL，PNUTS
- 書き込み最適化: Cassandra, HBase

この論文の評価結果より，高頻度に発生するセンサデータの書き込みリクエストに対応するためには，Cassandra または HBase が望ましいことが分かる．しかし，両者ともに読み込みに関して課題がある．

2.3 データストリーム管理システム

ユーザから送られてくるデータを一続きのストリームとしてとらえ，ストリームから必要なデータのみを取得するデータ駆動型のシステムをデータストリーム管理システム (Data Stream Management System, DSMS) という．代表的なものとして，Auroa[13]，Borealis[14]，StreamSpinner[15] などがある．DBMS と大きく違う点は，データの永続化を犠牲にして処理速度と応答性の向上を行っている点である．

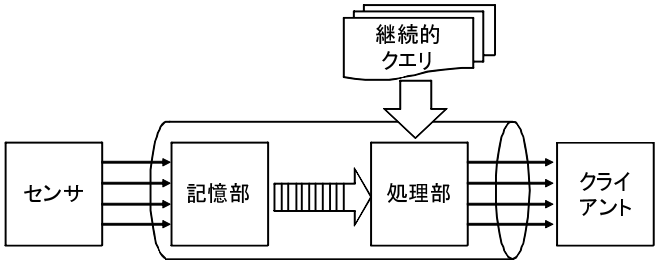


図 3 DSMS の構成

図 3 に示すように，DSMS は記憶部とクエリ処理部からなる．到着したデータは，DBMS ではディスクに保存されるが，DSMS ではメモリ上に保存し，一定の時間が経過すると破棄される．このように，DSMS はディスクアクセスを排除してメモリアクセスで処理を実行することにより，処理速度の向上を実現している．クエリ処理部では，ユーザから送られるクエリに従った処理を行い，必要なデータの抽出を行う．ただし，DBMS のクエリとは異なり，一度入力されたクエリはデータの到着に従い継続的に結果を返す継続的クエリとして振る舞う．ユーザは一度クエリを入力するだけで必要なデータを入手することが可能となり，DBMS のように定期的にデータを取りに行く必要が無くなる．

DSMS にデータ蓄積機能を加えようとする研究も行われている．Harmonica[16] は，DSMS のストリーム処理結果の出力先として RDB を加え，データ抽出・通知処理と並行してデータの蓄積を実現するシステムである．DSMS と DBMS を組み合わせるにあたり，DSMS の処理レートが DBMS の処理レートを上回った場合に処理が停止してしまう点が課題として考えられる．そこで Harmonica は，処理可能性の算出とそれを元にした処理の最適化により上記課題の解決を図っている．処理可能性の算出には，DSMS に到着したデータのレートと DBMS に書き込むレートを用いている．この 2 つのレートから，DSMS 内に構築されている処理プランにおける処理コストを求め，1 秒以内に収まれば処理可能，収まらなければ処理不可能と判断している．最適化は処理不可能と判断された場合に実行され，現在のプランにおいて優先順位に基づいた蓄積演算の結合を行い処理コストの軽減を図る．しかし，処理のレートが変化した場合に再計算が必要な点や，単純にデータ量が多いために処理不可能であるケースに対応できない点が課題である．

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

表 2 関連研究まとめ

名前	分散	安定	Write	過去データに関する Read	最新データに関する Read
P2P	○	×	-	-	-
DBMS(書き込み最適化)	○	○	○	△	×
DBMS(読み込み最適化)	○	○	△	○	△
DSMS	○	○	×	×	○
Harmonica	○	○	△	○	○
提案システム	○	○	○	△	○

2.4 関連研究まとめ

図 2 に関連研究のまとめを示す．P2P ネットワークは高い分散性を持つが，データの読み書きが安定しない．DBMS は高いデータの読み込みまたは書き込み性能を持つが，もう片方の性能が課題となる．DSMS は高い読み込み性能を持つが，データの蓄積が行えない．Harmonica は，DSMS に RDB を組み合わせたシステムであり，データ蓄積が可能であるが，単純に書き込まれるデータ量が多い場合に対応できない．

収集したセンサデータを利用したサービスについて考えた時，交通渋滞提供や高齢者の見守りサービスなどが考えられる．これらサービスによるデータの読み込み要求を分類すると，用途から大きく分けて次の 2 種類がある．

(1)　マイニング処理のための，過去データの読み込み
(2)　モニタリング処理のための，最新データの読み込み

(1) は 1 日に数回程度の低頻度な要求であるのに対し (2) は常に発生し続ける高頻度な要求である．ここでシステムの負荷を考えると，(2) への対応は重要となる．

センサデータ収集システムには，高頻度データ読み込み要求 (2) への対応に加え，高頻度なセンサデータ書き込み要求への対応も求められる．これは，表 2 における Write と最新データに関する Read に相当するが，両者を満たす研究は存在しない．本研究では，DSMS と書き込みに最適された DBMS を組み合わせ，高速な読み書き両方に対応できるセンサデータ収集システムを提案する．

3. 提案システム

3.1 概　　要

これまでの研究では，読み込みと書き込みどちらかに焦点を当てたシステムが考えられてきた．しかし，センサデータ収集システムの構築において，センサからの高頻度なデータ書き込みだけでなく，サービスからの高頻度なデータ読み込みの両方に対応できるスケーラブルなシステムを構築することが重要と考える．そこで本研究では，高頻度な読み込みが最新データを対象に行われている点に着目し，DSMS と書き込みに最適化された分散 DBMS を組み合わせたシステムを提案する．

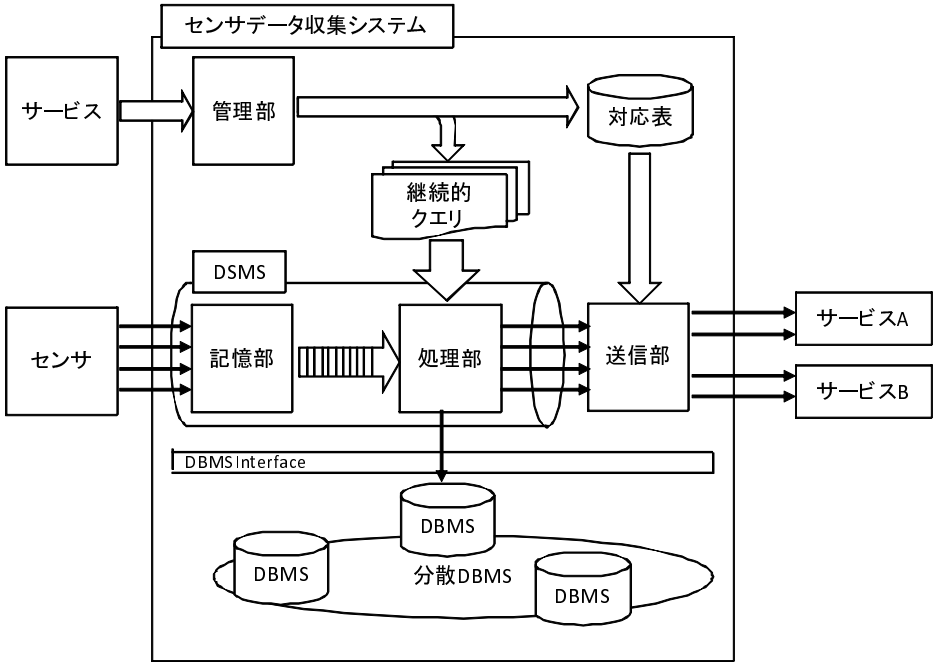


図 4 センサデータ収集システムの構成

提案するシステムでは，高頻度な読み込みを DSMS で処理し，DBMS に要求を伝えないことで読み込みと書き込みの分離を行う．そして，読み込みを DSMS に，書き込みを DBMS にそれぞれ任せることでシステム全体が高頻度な読み書き両方に対応できるようにする．また読み書きの分離は，最適化の手法を増やすことにも繋がる．例えば書き込み性能の最適化に関しては，データをある程度蓄積してから一気に書き込むバルクインサートを用いた手法が考えられる．一般的に，バルクインサートはネットワークの遅延の影響が少ない点やシステム側での最適化が行える点から，データを 1 つずつ挿入するシングルインサートよりも早いと言われている．高頻度な読み込みを排除した本システムでは，書き込まれてから読み

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

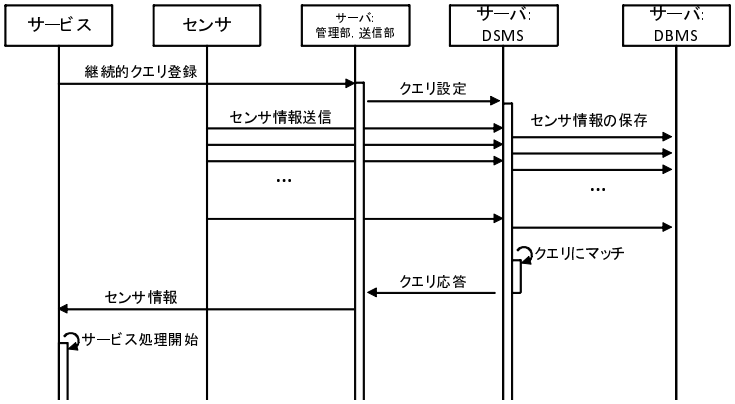


図 5 動作シーケンス

込まれるまでの時間が長くなり，時間制限が緩和される．バルクインサートを利用することで，従来のシステムよりも多くのデータを蓄える可能性を高め，より書き込みの効率化が期待できる．

図 4 に提案するセンサデータ収集システムを示す．センサデータ収集システムは DSMS と分散 DBMS，またサービスやセンサをシステムと結びつける管理部や送信部からなる．まず，サービスはデータを取得するための継続的クエリを管理部に対し送信する．管理部は DSMS へ継続的クエリを登録し，更に登録した継続的クエリと登録者の情報を対応表に保存する．次に，センサは収集したデータを DSMS に対して送信する．DSMS では，記憶部で一時的にメモリ上にデータが保存され，処理部で継続的クエリによるデータの抽出処理を行う．この際，分散 DBMS へデータの書き込みが並行して行われる．抽出されたデータは送信部へと送られ，対応表を元にクエリを設定したサービスを発見し，対応するサービスへデータの通知を行う．

図 5 に動作シーケンスを示す．例えば心拍などのバイタルデータによる高齢者見守りサービスを考えた時，システムは次のような動作を行う．まず，サービスは「各ユーザの心拍値が危険値を超えた場合に，ユーザの ID と心拍値が欲しい」という継続的クエリを管理部へと送信する．管理部は，クエリを DSMS に設定し，クエリとサービスの IP アドレスを対応表に保存する．次に，高齢者の持つ心拍計のセンサはデータを収集し，DSMS へ収集したデータの送信を行う．DSMS は，先ほど登録したクエリを用いて受信データの評価を行い，心拍値が一定の値を超えていた場合は送信部へデータを送る．送信部では，対応表からクエリを設定したサービスの情報を取得し，ユーザ ID と心拍値の通知を行う．見守りサービスは，このようにしてセンサデータ収集システムからデータを取得し，例えば関係者への警報メール送信等のサービスを起動する．なお，DBMS へのデータの書き込みは DSMS の評価結果に関わらず行われ，マイニング処理などに利用される．

4. 評　　価

提案システムが DBMS のみを用いて構築した従来のシステムに対して，どの程度の書き込み性能とスケーラビリティが得られるか比較評価により検証した．評価では，書き込み性能および読み込み要求による影響を詳細に分析するために，分散データストア用ベンチマーク YCSB を用いた．

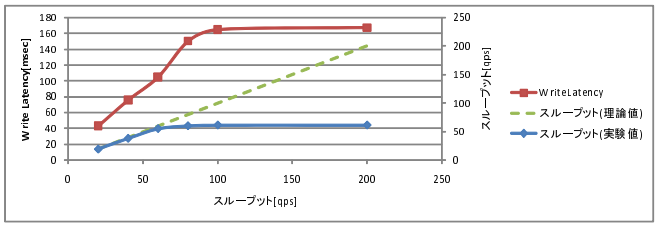
提案システムは NoSQL のうち最も書き込みに優れていると言われる Cassandra を，従来のシステムは RDB の一種である PostgreSQL を用いて構築を行った．なお，スケーラビリティの比較評価を行うため，従来システムでは PostgreSQL クライアントにシャーディング機能の実装を行った．シャーディングは，複数のノードそれぞれに対して自分が持つべきデータの範囲を決めておき，その範囲に従ってデータの挿入を行う手法である．これにより，テーブル 1 つ当たりのデータ量が減ると同時に分散環境にも適応できるようになる．今回の実装では，全てのデータベースの IP を配列に挿入し，キーのハッシュ値をノード数で割った余り番目のサーバを管理ノードとして選出するようにした．表 3 は各 DBMS を動かすノードのスペックであり，このノードを複数用いてクラスタの形成を行った．

評価に用いた YCSB は，前準備として対象の DB に対してある一定数のデータを DB にロードさせたのち，スループットを段階的に上げながら読み書きに関しての性能を測定する．事前に DB にロードさせる処理は，性能評価時に DBMS は保存されているデータの数による影響を受けるためであり，特に Cassandra の場合は全てのデータをメモリ上で処理させないようにするためである．性能を測定する際には，リクエスト全体における読み込みと書き込みの比率を変更する事ができる．

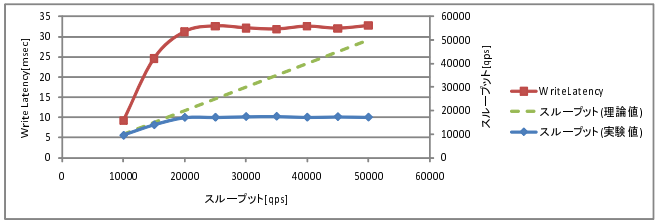
4.1 書き込み性能評価

提案システムには，センサから送られる大量のデータ書き込みリクエストの処理が要求され，高い書き込み性能とスケーラビリティが求められる．そこで YCSB を用いて，書き込み要求に対するスループットを計算し，書き込みに最適化された Cassandra でセンサデータ

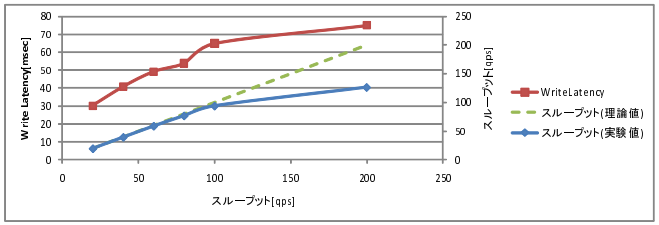
Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10



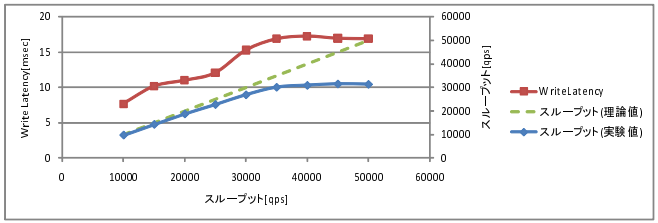
(a) PostgreSQL ノード 1 台



(b) Cassandra ノード 1 台



(c) PostgreSQL ノード 3 台



(d) Cassandra ノード 3 台

図 6 書き込み性能

表 3 ノードスペック

マザーボード	GIGABYTE H55M-UD2H
OS	Ubuntu 9.10
CPU	Intel(R) Core(TM) i5 CPU 661 @ 3.33GHz
RAM	4.00G
HDD	1T SATA

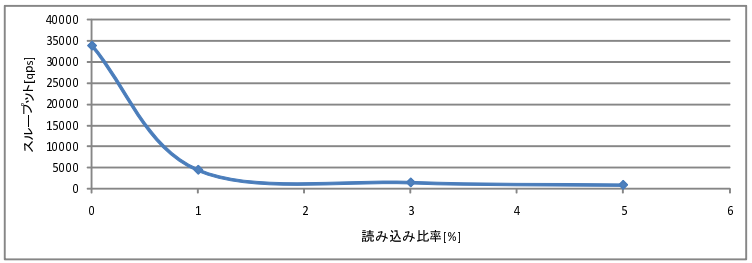
の書き込みに対してどの程度対応できるかの測定を行った．また，クラスタのノード数を増加させたときのスループットの変化から，スケーラビリティの測定も行った．

図 6(a)，図 6(b) はノードが 1 台の時の書き込み遅延，スループットの理論値と実験値を示している．DBMS のスループットは図の理論値のように徐々に上昇するはずであるが，ある点を境に理論値と実験値に差が生じはじめ，やがて安定する．この安定した点の値は，DBMS を構成するノードのソフトウェアないしハードウェア上のボトルネックによる性能限界値に達していると考えられる．図より，Cassandra の方がスループットが高く，PostgreSQL の 61qps に対して Cassandra は 17,437qps と約 286 倍の性能差が生じている．

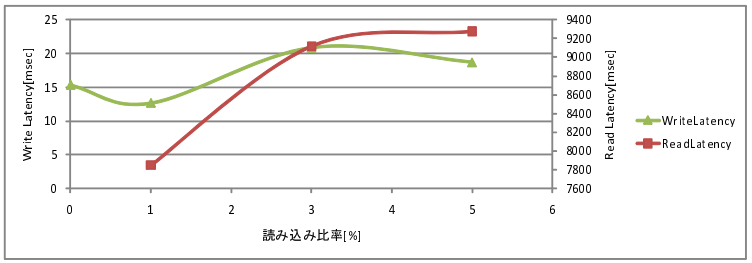
図 6(c)，図 6(d) はノード 3 台でクラスタを形成した場合の評価結果を示している．図 6(a)，図 6(c) より，PostgreSQL はノードを 1 台から 3 台へ変更することでスループット 61qps から 126qps と変化しており，性能が約 2 倍向上している．一方，図 6(b)，図 6(d) でも同様の変化が発生している．Cassandra はノード 1 台から 3 台への変更によりスループットが 17,437qps から 30,810qps と変化し，性能が約 1.8 倍向上している．PostgreSQL に比べ Cassandra の方が性能向上率が低いのは，ノード間での P2P 通信によるオーバーヘッドが影響していると考えられる．

今，64,500qps の書き込みが発生する高齢者見守りサービスを構築したとする．この負荷に Cassandra を用いて対応するためには，ノード 10 台程度でクラスタを形成することで十分対処することができる．PostgreSQL で対応する場合は，2000 台ものノードでクラスタを形成する必要がある．また更に，PostgreSQL は分散環境を構築するためにシャーディングを用いているため，ノードを追加する際にクラスタ全体に影響を与える可能性がある．例えば今回のような実装方法では，ノードを 1 台追加すると全てのデータの管理ノードが 1 つずれるため全データに対して再配置を行う必要がある．Cassandra の場合はデータのキーのハッシュ値で管理ノードを決めるコンシステント・ハッシュ法を用いているため，ノード追加による影響を局所化できる．コンシステント・ハッシュ法は，ノードにもハッシュ値を設定し，挿入する際にキーのハッシュ値が最も近いノードを管理ノードとして選ぶ手法であ

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10



(a) Throuput



(b) Latency

図 7 読み込み要求の変化による性能の変化

る．シャーディングのようにノード数で割る必要が無く，ノードを追加する場合の影響は挿入先の前後のノードに対する再配置のみで収まる．

4.2 読み込み性能評価と最適化

センサデータを書き込む際，読み込み要求を並列して処理することで性能が低下してしまう状況が考えられる．提案システムでは高頻度な読み込み要求の分離を行っているが，どの程度の頻度を分離する必要があるかの測定を行った．また，高頻度な読み込み要求の分離によりどの程度バルクインサートによる性能向上が見込めるかの測定も行う．

図 7 は，読み込みの比率を 0%から 5%と変化させたときの，Cassandra のスループットと遅延時間を示す．図より，読み込み要求の比率の増加に従いスループットが低下していることが分かる．特に，読み込み比率が低い範囲においてはスループットの減少が顕著であり，比率 0%と 1%を比べた場合は 1/8 倍となっている．これは，読み込みの方が書き込みよりも圧倒的に処理に時間がかかるためであり，そちらにノードのリソースが取られているためである．以上より，1 秒の内に何度も読み込みが発生するケースでは著しく性能を低下させ，このようなケースは高頻度読み込みであるとして DSMS による分離が必要となる．

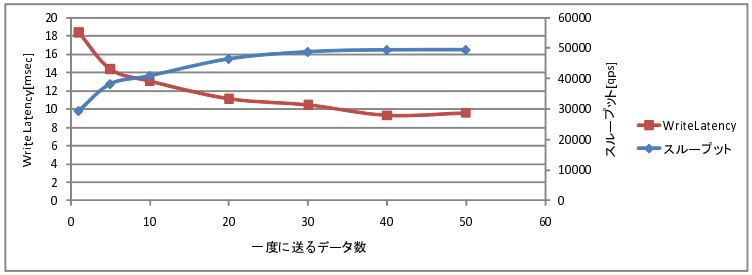


図 8 バルクインサートによる性能の変化

図 8 は，到着したデータをある程度蓄積してから書き込むバルクインサートを行った場合のスループットと遅延時間を表す．スループットは蓄積するデータ数に応じて上昇するが，データ数 20 付近で安定を見せる．この時点でスループットは 46,448qps となっており，蓄積しない場合に比べて性能が約 1.5 倍向上している．以上より，提案システムにバルクインサートを適用することで最大で 1.5 倍の性能向上が見込める．この性能向上率は，Cassandra のノードを 3 台追加するのに相当する．

5. ま と め

本研究では，DSMS と書き込みに最適化された分散 DBMS を組み合わせたセンサ情報収集システムの検討，ならび性能評価を行った．従来のシステムでは，高頻度読み込みと書き込みの両方が DBMS に集中していた．本システムでは，この高頻度な読み込みと書き込みを分離し，読み込みは DSMS に，書き込みは分散 DBMS にそれぞれ対応させる．これにより，高頻度な読み書き両方への対応とスケーラビリティの確保を行う．

評価では，RDB の一種である PostgreSQL と NoSQL の一種である Cassandra を用いた書き込み性能の比較評価，ならび読み込みの頻度の変化やバルクインサートの有無による性能への影響を測定した．比較評価では，一般的に言われているように Cassandra の方が書き込み性能に優れており，高頻度なセンサデータの書き込みへの耐性が高いことを確認した．また，ノードの追加による負荷への対応ができる高いスケーラビリティを持っており，64,500qps の書き込みが発生する高齢者見守りサービスを想定したとしても十分に対応する事が出来る．読み込み頻度による評価では，1 秒の間に何度も読み込み要求が発生するよう

Vol.2011-DPS-146 No.10
Vol.2011-CSEC-52 No.10
2011/3/10

なケースは高頻度読み込みであるとし，DSMS による分離が必要であることを確認した．バルクインサートでは最大で約 1.5 倍の性能向上が見込め，高頻度な読み込み要求の排除により最大値に近い性能向上率が期待できる．今後の予定としては，DSMS も含む全体としての評価ならびボトルネックの発見と解消が考えられる．

謝辞 本研究の一部は，科学研究費若手研究 A（21680007）の助成によるものである．ここに記して謝意を表す．

参 考 文 献

1) 飯島護久, 堀口良太. プローブデータに基づくエリア流動性情報提供に関する研究. 第 9 回 ITS シンポジウム 2010 予稿集, pp.1-4, 2010.
2) 黒田 晋矢, 岡本 章裕, 横山 昌平, 福田 直樹, 石川 博. センサデータを利用したフリーアドレスオフィス環境における省電力化のための人物誘導システムの試作. 電子情報通信学会 第 1 回データ工学と情報マネジメントに関するフォーラム (DEIM2009) (2009).
3) 塚本吉俊, 高松周一, 森武俊. 安心・安全のための移動体センシング技術-高齢者異常検知予測システムの開発-. 富山県工業技術センター研究報告, 2010.
4) 平成 22 年版　高齢社会白書. http://www8.cao.go.jp/kourei/whitepaper/w-2010/zenbun/22pdf_index.html.
5) I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In Proc. ACM SIGCOMM, pp.149-160, 2001.
6) B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, 2001.
7) P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In Proceedings of IPTPS, pp.53-65, 2002.
8) J. Aspnes, and G. Shah. Skip Graphs. In ACM Transactions on Algorithms, Nov. 2007.
9) B. Cooper, A. Silberstein, E. Tam, R. and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In Proceedings of the 1st ACM symposium on Cloud computing (SoCC'10), 2010.
10) A. Lakshman, and P. Malik. Cassandra: a decentralized structured storage system. In 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS'09), 2009.
11) HBase, http://hbase.apache.org/.
12) B. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H. Jacobsen, N. Puz, D. Weaver and R. Yerneni. PNUTS: Yahoo!ś hosted data serving platform. In Proc. 34th VLDB, 2008.
13) D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul and Stan Zdonik. Aurora: a new model and architecture for data stream management. The International Journal on Very Large Data Bases, 2003.
14) D. Abadi, Y. Ahmad, M. Balazinska, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, E. Rasin, E. Ryvkina, N. Tatbul, Y. Xing and S. Zdonik. The design of the borealis stream processing engine. Second Biennial Conference on Innovative Data Systems Research (CIDR'05), Jan, 2005.
15) StreamSpinner, http://www.streamspinner.org/.
16) 山田真一, 渡辺陽介, 北川博之, 天笠俊之. データストリーム管理システム Harmonica の設計と実装. 情報処理学会論文誌：データベース, Vol.48, No.SIG14 (TOD35), pp. 91-106, 2007.