Vol.2013-DPS-157 No.5
2013/10/17

M2M データ解析手法の検討

安田晃久[†1]　北上眞二[†2]

機器間の相互通信／制御によって，人手を介さずシステム化する技術を M2M（Machine to Machine）と呼ぶ．M2M で機器を制御する場合，稼動状態把握のための解析処理が必要だが，解析のために大量データを逐次リモートサーバへ送信するようなシステム構成を取ると，通信負荷が高まり，制御実行までのタイムロスが大きくなるという課題がある．これを解決するには，ローカル環境でデータ解析を行うことが有効である．本稿では，組み込み機器上で機器データ解析に有効なリレーションシップを表現する手法について述べる．

Consideration of M2M Data Analysis Method

AKIHISA YASUDA[†1]　SHINJI KITAGAMI[†2]

M2M(Machine to Machine) system is an optimized autonomous system constructed by communication and control among devices without human intervention. In M2M system, it is necessary to analyze data collected by the center server for controlling devices, but it increases the communication traffic and causes the delay of device control. Therefore, it is effective to analyze data in the device side. This paper attempts to show a method which represents the relationship of data using the graph database, to make data analysis easy in devices.

1. はじめに

機器同士をネットワークで繋げ，機器が所有するデータを交換し合い，または当該データを基に機器の動作を変更することで，人手を介さず自動的にシステム化する技術を M2M（Machine to Machine）[1]と呼ぶ．近年の通信インフラの整備や組み込み機器の高度化を受け，M2M システム構築化の敷居が下がり，M2M を用いたサービスの提供や，M2M システム標準仕様の策定が進んでいる[2][3].

M2M では有線，無線を問わず，ネットワークで繋がる機器全てを対象とするため，M2M システム上には多数の機器が存在し，それに伴い，取り扱うデータサイズやデータ種別も増加することとなる．例えば，温湿度センサのように，逐次温度と湿度のログデータを生成する機器の場合，室内環境を解析するためには複数箇所にセンサを設置する必要があり，それだけ蓄積するデータサイズも大きくなる．また，省エネのために機器の稼動を制御しようとした場合，分電盤やコンセント等の電力供給元から収集する電力量データだけでなく，制御実行可否を判断する材料として，機器毎に固有の稼動情報を収集する必要がある．これにより，制御対象の機器数だけデータ種別が増えることとなる．

近年，こうした大量／多種のデータを“ビッグデータ”[4]と呼び，ビッグデータ解析の手段として，NoSQL（Not Only SQL）[5]と呼ばれるデータ管理手法が提案されている．M2M データについても，データサイズやデータ種別の増加を伴う場合，NoSQL による解析を行うのが適切である．

しかし，M2M データの解析に NoSQL を適用しようとした場合，データ解析処理実行環境に伴う課題が生じる．NoSQL ではデータ解析処理をスケールアウトさせるため，データ解析専用サーバを多数備えた，分散処理システムを前提としている．そのため，例えば工場等に設置された M2M データ収集を行う組み込み機器と，データセンタ内に設置されたデータ解析サーバのように，データ収集者とデータ解析者が物理的に離れた環境下でデータ解析処理を行う場合，ネットワーク経由で大量のデータをデータ解析サーバへ送信する必要があり，通信負荷増大，延いてはデータ送信からデータ解析終了までの遅延時間の増加を招く．ポータビリティの点から，M2M 機器には通信手段として無線通信が適用されるケースが多く，この場合，遅延の傾向は更に顕著となる．

従って，M2M データ解析の目的がシステム最適化のための機器制御であるならば，遠隔地にあるデータ解析サーバへデータを送信せず，データ収集を行う組み込み機器内でデータ解析処理を行うことが望ましい．しかし，このような組み込み機器をターゲットにデータ解析処理を実行する基盤システムについて，一般化されているものはない．

そこで，本稿では組み込み機器を対象に，データ解析処理に必要な関係性システムの構築に要する構成を提案する．

2. M2M データ解析における課題

2.1 NoSQL

従来からシステムのデータ管理に用いられてきたものに RDBMS（Relational Dababase Management System : 関係データベースシステム）がある．これに対し，近年，ビッグデータを容易に取り扱うことや，Web サービスシステムとシームレスに結合することを目的に提案されている手法が

†1 NPO 法人 M2M 研究会
Study Group on M2M
†2 三菱電機（株）情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation

NoSQL である．

RDBMS と NoSQL を比較すると，データ管理の要件が異なる．RDBMS では CAP 定理（Consistency：一貫性，Availability：可用性，Partition Tolerance：分割耐性）[6]のうち，一貫性と可用性の両立を重視した設計となっており，ACID（Atomicity：原子性，Consistency：一貫性，Isolation：独立性，Durability：耐久性）[7]を遵守するよう動作する．そのため，一般システムでは特に，データサイズが過度に大きくなるとディスク IO がボトルネックとなり，データの取り扱いが困難となる．また，RDBMS はスキーマに依存した構成となっているため，データ種別の変更や追加といった操作に容易に対応することが出来ない．

これに対し，NoSQL では可用性と分割耐性を両立する BASE（Basically Available, Soft-state, Eventually consistent）[8]に順じた構成となっている．そのため，厳密な整合性を保証することは出来ないが，ビッグデータのような，大規模分散処理や多様なデータに容易に対応することが出来る構成となっている．

即ち，RDBMS と NoSQL は適応要件が異なるため，用途に応じて選択すべきものとなっている．

NoSQL に分類されているデータベースには主に，KVS（Kye Value Store），列指向型データベース，ドキュメント指向型データベース，グラフデータベースの 4 種類が存在する．これらは全て，分散処理性能向上のために KVS の機能を保持しつつも，各々のデータ管理目的に応じた構造，機能となっている．

列指向型データベースでは，スキーマレスという点では KVS と同様であるが，行毎に値の形式を不定形とし，同じ行キーに対して異なる値（この場合，列キーを別途指定する）を追記出来る点が異なる．

ドキュメント指向型データベースでは，文字列データとして構造化した，スキーマレスな KVS データを文字列のまま操作することが出来る．そのため，例えば Web サービスシステムと連携して動作させる場合，Web サービスでは文字列データを取り扱う頻度が非常に高いため，他のデータベースと比較して容易に連携させることが出来る．

グラフデータベースでは，データベース上に各データの関係性に係わる情報を保持する．そのため，関係性の追加／変更／削除や単一始点最短路問題など，グラフ理論に係わる操作について，容易に対応することが出来る．

また，NoSQL を用いた分散処理では大きく分けて，過去データを扱うバッチ処理と，現在データを扱う CEP（Complex Event Processing：複合イベント処理）に二分される．バッチ処理では，予め蓄積されたデータに対し，任意のタイミングで処理を実行する．CEP では，新規に記録されたデータに対し，逐次処理を実行する．これらはデータ解析実行のタイミングや同期のタイミングに相違があるものの，データ解析要件として，一元管理されたログデータを対象とすることが共通している．

2.2 M2M データ解析

M2M システムで管理される M2M データがビッグデータと同義となる場合，そのデータ管理手法として，NoSQL を適用するのが妥当である．

図 1 に NoSQL データシステムを用いた M2M システムの構成図を示す．このシステムは，遠隔地に設置された組み込み機器のログデータを収集／解析するものであり，データ解析の目的は機器を制御することである．

図 1 の中で，データセンタ内に設置されたデータ解析クライアントとデータ解析サーバ，そしてデータシステム・マスタサーバは，分散データ解析処理専用システムを構成しており，データ解析サーバで共有されている NoSQL データシステム（このシステムは別途マスタサーバが管理する）に対し，データ解析クライアントから命令されたデータ解析ジョブを並列処理で実行し，データ解析結果を再び NoSQL データシステムに格納することを示している．

また，ネットワーク経由でデータセンタ内の NoSQL データシステムへログデータを送信する組み込み機器は，複数台の機器，例えば温湿度センサ等，を子機として束ねる親機として機能し，子機から収集したログデータを内部ストレージに集積し，適切なタイミングで NoSQL データシステムへ送信する．

こうした機器制御を前提とする M2M システムのデータ解析処理において，以下の点が課題として挙げられる．

- ログデータの収集からデータ解析終了まで処理が多岐に渡り，各処理の何れかがオーバーヘッドとなることで，制御実行までの間に遅延が発生し，最適なタイミングで制御を実行することが困難となる．

遠隔地に設置されている親機が，例えば 3G や LTE（Long Term Evolution）などの無線通信でネットワークに接続している場合，設置場所によって通信品質に相違が生じることが予想される．従って，例えばログデータの集積をトリガーとしてデータ解析を実行する場合，遅延を発生させずにデータ解析処理を終了させることが出来る，という保証は無い．

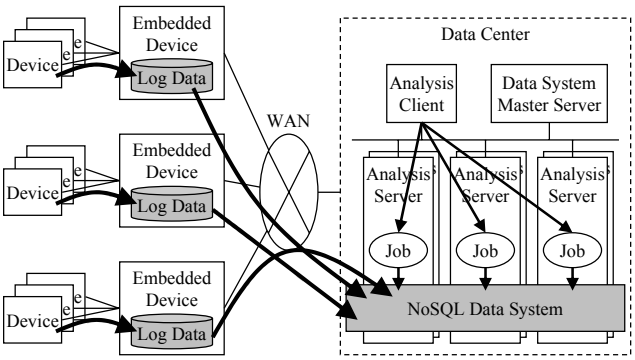

図 1　既存 NoSQL を用いた M2M システム構成図

2.3 組み込み機器上のデータ解析要件

遠隔地に設置された機器のログデータ解析では，ログデータの収集から解析終了までの間に遅延が発生し，最適なタイミングでの制御が困難となるケースが考えられる．そこで，データ解析処理をサーバ上で実行せず，ログデータ収集を行う組み込み機器上で，直接データ解析を実行する場合の要件と構成を考える．

図 2 に組み込み機器上でログデータ解析処理を行う際の構成を示す．ここでは図 1 と同じく，組み込み機器を親機とし，その配下に複数の子機が接続される構成としている．図 2 を基に，要件の対応を以下に示す．

- ログデータの管理：
  組み込み機器上のログデータを単なるファイルとして管理した場合，操作対象のデータにアクセスする手段が限定的となるため，データ解析処理が非常に困難となる．そのため，組み込み型のストレージエンジンを利用するのが適切である．また，データサイズに依存せず，適切な時間で CRUD（Create：作成，Read：読み出し，Update：更新，Delete：削除）を行うことを考慮すると，RDBMS ではなく NoSQL を選択するのが適切である．
- ログデータシステム：
  一般的に，データ解析モデルで利用するデータは，データ管理システムによって一元管理されていることを前提としている．しかし，組み込み機器上のログデータは分散管理されているため，他の組み込み機器で集積されているログデータを，ローカルで集積しているログデータと同様に取り扱うことが出来ない．組み込み機器上でデータ解析処理を行うならば，分散管理されているログデータを統括的に取り扱うための仕組みが必要である．
- 分散データ解析処理：
  各組み込み機器内で実行するデータ解析は並列処理となるため，例えば同じ解析結果データを異なる組み込み機器で共有する必要がある場合，2 つの組み込み機器の間で同期を取るための仕組みが必要である．

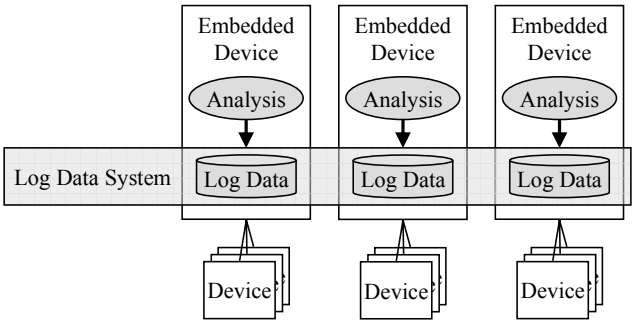

図 2　組み込み機器上のデータ解析システム構成

3. 組み込み機器上のデータ解析システム

組み込み機器上のデータ解析要件に対し，各項目に対する提案手法を提示する．

3.1 ログデータの管理

オープンソースの組み込み型 NoSQL（列指向型データベース）として LevelDB[9]がある．LevelDB はライブラリ形式で利用可能，シャーディングによる効率的なデータ管理，データ格納にファイルを使用，という特徴を持っている．そのため，リソース制限の厳しい組み込み機器上で動作させるのに適した NoSQL であると言える．

LevelDB に対し，組み込み機器上で利用される組み込み型 RDBMS と比較すると，ログデータの管理において，以下の点で有効である．

- LevelDB ではデータ検索の際に結合処理が不要なためオーバーヘッドが小さく，組み込み型 RDBMS と比較して，高速検索が可能[10]
- LevelDB ではスキーマレスに列データを追加出来るため，スキーマ依存な組み込み型 RDBMS と比較して，行データの検索性能を落とすことなく容易に収集データの追加／変更が可能
- LevelDB では段階的な自動シャーディングとデータ圧縮を行うため，組み込み型 RDBMS と比較して，効率良くディスク領域を利用することが可能

但し，ログデータの冗長化については LevelDB でも，組み込み型 RDBMS でも，ストレージエンジン側では保証していない．そのため，ログデータの冗長化が必要な場合，別途アプリケーション側で対応する必要がある．

3.2 ログデータシステム

異なる組み込み機器の間でログデータを参照したり，取得するログデータシステムを考えると，共通プロトコルによるネットワーク上のメッセージング処理が必要である．また，組み込み機器の故障によるシステム上の SPOF（Single Point of Failure：単一障害点）を避けるためには，ログデータシステム上のログデータは，P2P（peer to peer）ネットワークのようなオーバーレイネットワーク上のアドホックな構成で接続する必要がある．

図 3 に，組み込み機器上のログデータと，当該データを基に構築するオーバーレイネットワークの関係を示す．図 3 に示す通り，本稿ではログデータとは別に，データをノードとして定義するグラフデータベースを設け，当該グラフデータベース間のオーバーレイネットワークにより，データ間の関連性を表現する手法を提案する．

グラフデータベースではノード間の関係性を隣接リストで保持し，当該隣接リストから，グラフ構造に基づいた性質を計算する．そのため，組み込み機器上のデータ解析処理側は，API 経由で自己が管理するグラフデータベースを操作するだけで，ログデータシステム全体で相互に関連し

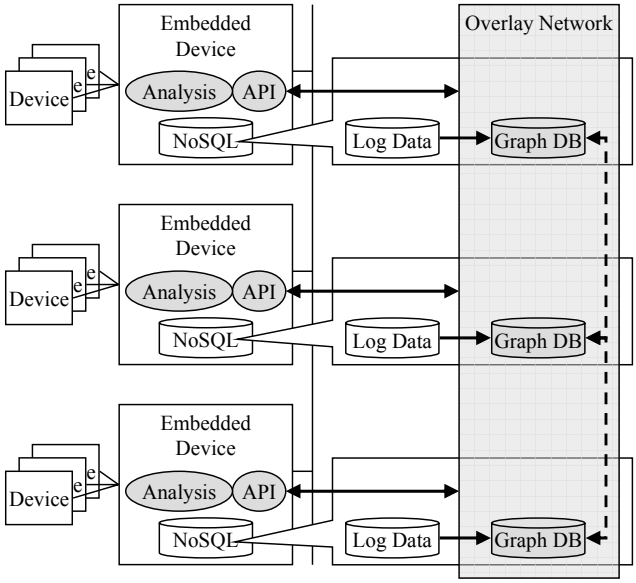

図 3　仮想ネットワークによる M2M データ解析システム

合う関係を表現することが出来る．また，図 3 のように，データが分散管理されている環境でデータ間の関連性を参照する場合，元のログデータを直接参照するより，グラフデータベースとして予め解析されたデータを利用する方が効率が良い．

但し，グラフデータベースではシステム管理機能がシステム全体のデータベースを一元管理していることを前提としており，図 3 のように，分散管理された複数のグラフデータベース間を仮想的に接続するような構成は一般的ではない．従って，本稿で提示した構成のグラフデータベースを利用する場合，別途新規開発が必要である．

また，一般的にデータ解析モデルでは，システム内のデータ位置（任意の機器とデータを一意に紐付けた情報であり，例えば，建物 2 階に設置された空調機固有の消費電力量，のような情報を指す）を予め把握していないと，データ間の関連性を考慮したデータ解析処理を実行することが出来ない．そのため，本提案構成では，組み込み機器内のデータ探索処理は不要である．

3.3 分散データ解析処理

図 3 に示した通り，組み込み機器上のデータ解析処理は並列で実行される．そのため，例えば，ある組み込み機器上のデータと他の組み込み機器上のデータで関係性を同期して更新したい場合，API 経由でグラフデータベースへの操作をロックすればトランザクションを構築することが出来，システム上は ACID 特性を保証することが可能である．この仕組みを図 4 を用いて説明する．

(A) 組み込み機器 A から組み込み機器 B に対し，各々が管理するデータ（ノード）間の関連性を変更するためのリクエストメッセージをネットワーク経由で送信する．組み込み機器 A のグラフデータベースはこの時点でロックされる．

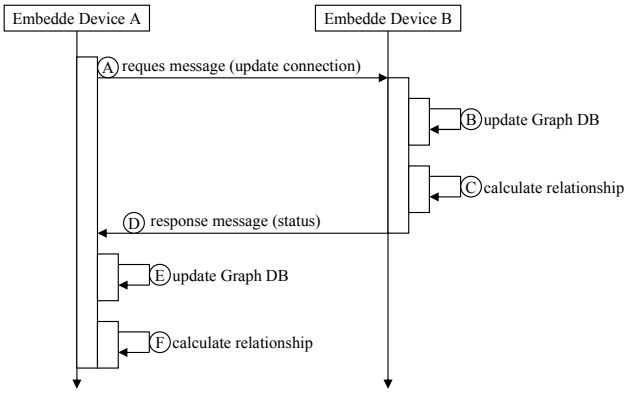

図 4　グラフデータベースの同期処理

(B) 組み込み機器 B がリクエストメッセージを受信すると，メッセージに合わせてグラフを変更する．また，組み込み機器 B が管理するグラフデータベース操作のロックを開始する．

(C) 組み込み機器 B はグラフデータベース内で変更したグラフのトポロジに合わせ，データ解析処理に必要な経路長等の計算を実行する．

(D) 組み込み機器 B はグラフデータベースのロックを解除し，操作結果をレスポンスメッセージとして送信する．

(E) 組み込み機器 B からレスポンスメッセージとして送信されてきた操作結果に合わせ，組み込み機器 A は，自己の管理するグラフデータベースのグラフを変更する．

(F) 組み込み機器 A は変更したグラフトポロジに合わせ，データ解析処理に必要な計算を予め実行しておく．組み込み機器 A と組み込み機器 B ではグラフデータベースで管理する関連性が異なるため，(B)の結果を利用することは出来ない．計算処理が終了した時点で組み込み機器 A のグラフデータベースのロックを解除する．

但し，同期処理が多数のグラフデータベースに跨って実行される場合，他の組み込み機器上のデータ解析は処理を進めることが出来なくなってしまうため，遅延が発生する．本稿の目的が，データ解析処理における遅延を抑止することである以上，遅延時間を増加させる手段の採用は必要最低限とすべきである．また，グラフデータベースのロック解除についても，メッセージが確実に通知出来る仕組みを構築しなければデッドロックに陥ってしまう．そのため，分散処理環境で利用する場合は，例えば規定時間に応じて自動的にグラフデータベースのロックが解除される仕組みを導入する必要がある．従って，本稿で提案する構成では，大規模な同期処理を必要とするデータ解析モデルではなく，結果整合性[11]を許容するデータ解析モデルに適用するのが望ましい．

4. まとめ

本稿では，M2M システムのデータ移送，及びデータ解析に要する時間を短縮するため，M2M システムの構成要素として設置される組み込み機器上で，直接データ解析処理を実行するための基盤システムを提案した．今後は，提案方式を実装したシステムを用いて，データ解析処理に要する時間を計測し，その有効性を評価する予定である．

参考文献

1) D. Boswarthick, O. lloumi, and O. Hersent: “M2M Communications: A Systems Approach”, Wiley, ISBN: 978-1119994757(2012)
2) 辻秀一，澤本潤，清尾克彦，北上眞二，“M2M（Machine-to-Machine）技術の動向 ”，電気学会論文誌 C，Vol. 133 No.3(2013)
3) ETSI, "Machine-to-Machine communications (M2M);Functional architecture", ETSI TS 102 690 V1.1.1(2011)
4) James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung Byers, “Big data: The next frontier for innovation, competition, and productivity”, http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation(2011)
5) Lith, Adam, Jakob Mattson, "Investigating storage solutions for large data: A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data", http://publications.lib.chalmers.se/records/fulltext/123839.pdf(2010)
6) Nancy Lynch and Seth Gilbert, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services", ACM SIGACT News, Volume 33 Issue 2, pg. 51-59.(2002)
7) Eswaran, K. P. Gray, J. N. Lorie, R. A. Traiger, I. L.. "The notions of consistency and predicate locks in a database system". Communications of the ACM 19 (11): 624–633(1976)
8) Eric Brewer, "Towards Robust Distributed Systems", http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf (2000)
9) Google LevelDB
http://code.google.com/p/leveldb/
10) LevelDB Benchmarks
http://leveldb.googlecode.com/svn/trunk/doc/benchmark.html
11) Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon's Highly Available Key-value Store”, SOSP'07(2007)