社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

スマートシティにおける大規模住宅ログのための体現ビュー生成サービスの実装

伊勢　勇輝† 　山本晋太郎† 　柗本　真佑† 　佐伯　幸郎† 　中村　匡秀†

† 神戸大学　〒 657–8501 兵庫県神戸市灘区六甲台町 1–1
E-mail: †{ise,shintaro}@ws.cs.kobe-u.ac.jp, ††{shinsuke,masa-n}@cs.kobe-u.ac.jp, †††sachio@carp.kobe-u.ac.jp

あらまし　我々の研究室ではスマートシティやスマートハウスから得られる，多種多様かつ大規模なログデータを扱うためのデータプラットフォーム Scallop4SC を提案している．Scallop4SC では，効率的な住宅ログの処理と利活用を支援するために，体現ビューのアイデアを取り入れた，MVaaS（Materialized View as a Service）を提供する．本研究では Scallop4SC 内での MVaaS の実現を目的として，MVaaS のアーキテクチャに関する検討とその開発に取り組む．提案する MVaaS では，住宅ログの検索条件や加工方法に関するデータ仕様書に基づいて，ログデータを処理するための MapReduce プログラムを自動生成する．生成された MapReduce を実行することにより，大規模データの中からアプリケーションが必要とするデータだけを体現ビューとして保持することが可能となる．
キーワード　住宅ログ，体現ビュー，MVaaS，MapReduce，KVS，HBase

Implementing Materialized View as a Service for Large-Scale House Log in Smart City

Yuki ISE†, Shintaro YAMAMOTO†, Shinsuke MATSUMOTO†, Sachio SAIKI†, and Masahide NAKAMURA†

† Kobe University, Rokkodai 1–1, Nada, Kobe, Hyogo, 657–8501 Japan
E-mail: †{ise,shintaro}@ws.cs.kobe-u.ac.jp, ††{shinsuke,masa-n}@cs.kobe-u.ac.jp, †††sachio@carp.kobe-u.ac.jp

Abstract　We have proposed a logging platform, called Scallop4SC(Scalable Logging Platform for Smart City), for managing various and large-scale log data from smart houses within a smart city. Moreover, we design that Scallop4SC provides MVaaS(Materialized View as a Service), which supports processing and using house log efficiently. In this paper, we implement an architecture of MVaaS using MapReduce on Hadoop and HBase KVS. Concretely, MVaaS automatically generates MapReduce programs with a spec for processing log data. The spec are drawn based on retrieval conditions or processing method for house log. By treating MVaaS, a developer of an application can dynamically create and efficiently access smart city data by only describing data specification for the application.
Key words　House log, Materialized view, MVaaS, MapReduce, KVS, HBase

1. はじめに

スマートシティ [1], [2] とは，ICT 技術を用いて都市全体の高度な効率化を目指す次世代都市計画である．スマートシティでは，都市内の様々なモノやシステムから情報を集め，その情報に基づいて都市の現状を把握し，状況に応じた付加価値サービスが提供される．こうした情報は，都市内に設置されたセンサやシステムのロガー等から収集される．例えば，スマートハウスから取得される電力や家電機器の使用状況などの住宅情報や，環境センサから取得される気温や湿度などの環境データ，道路や鉄道から収集される交通データなどが知られている．

先行研究 [3] では，スマートハウスから得られる様々な情報（以降，**住宅ログ**と呼ぶ）を扱うためのプラットフォーム **Scallop4SC**（**Scalable Logging Platform for Smart City**）を提案した．Scallop4SC では，多種多様かつ大規模な住宅ログを効率的に収集・管理するために，分散処理基盤 Hadoop と分散データベース HBase を利用しており，MapReduce プログラムを用いて住宅ログの処理が行われる．Scallop4SC の利用者は，Scallop4SC の提供する API を通じて様々な用途のアプリケーションやサービス（例えば，消費電力の可視化，無駄の振り返

りなど）に住宅ログを利用することができる．

さらに先行研究 [4] では，より効率的な住宅ログの処理を実現するために，**体現ビュー（Materialized View）**のアイデアを取り入れた **MVaaS（Materialized View as a Service）**を提案した．一般に住宅ログを活用するアプリケーションは，蓄積された 1 次データ（生データ）をそのまま利用するのではなく，アプリが必要とするデータを検索し，必要に応じて加工・整形してから利用する．しかしながら，スマートシティから収集される 1 次データは大規模なため，検索や加工に非常に時間がかかる．MVaaS を利用し，各アプリが必要とする住宅ログをあらかじめ検索・加工し体現ビューの形で保持しておくことで，効率的なデータ利用を促すことができる．

本研究では Scallop4SC における MVaaS の実現を目的として，MVaaS の実現方法についての検討とその開発を行う．具体的には，MapReduce プログラムの自動生成とそのプログラムの実行による自動的な体現ビューの生成により実現される．まず MVaaS の利用者は，アプリが必要とする住宅ログの検索条件や加工方法に関する仕様（**Data spec.**）を記述し MVaaS に登録する．MVaaS は登録された Data Spec. に基づいて，体現ビューを生成するための MapReduce プログラムを自動的に生成し実行する．生成された体現ビューは MVaaS の提供する API を通じて利用することができる．

MVaaS の効率を確かめるための評価実験として，3 種類の体現ビューを生成する Data Spec. を用意し，体現ビュー生成にかかる時間を計測する．さらに，体現ビュー自動生成による開発コストの削減効果を確認するために，Data Spec. の行数と，その Data Spec. と等価な処理を行う MapReduce プログラムの行数を比較する．実験の結果，体現ビューの自動生成が現実的な時間で処理可能であること，及び，体現ビュー自動生成によって大幅に開発コストが削減できることが確認できた．

2. 準　　備

2.1 スマートシティにおける付加価値サービス

スマートシティでは，都市内の情報を集めて都市の現状を把握し，状況に応じた付加価値サービス（**スマートシティサービス**と呼ぶ）が提供される．期待されるサービス分野としては，例えば，省エネルギー，交通の最適化，局地的な経済トレンド分析，エンターテインメント，地域医療・健康，災害対策，農業支援など多岐にわたる．

また各サービス分野内でも様々なバリエーションのサービスが考えられる．例えば，省エネルギー分野に焦点を当てると，次のようなスマートシティサービスが考えられる．

（1） **消費電力可視化サービス [5]:** スマートシティ内の住居（スマートハウス）から，消費電力情報を取得し，都市の電力使用状況を見える化するサービス．各住居単位や町単位，機器単位，現在消費電力量，過去の消費電力量の推移など，様々な観点から可視化する．実際に使用した消費電力を目に見える形で可視化することにより，住民の省エネ意識の向上をはかる．

（2） **無駄検出サービス [6]:** スマートハウス内の消費電力やセンサのデータから，無駄な電力使用がないのかを検出し，通知するサービス．また, 過去にどのような無駄があったのかを振り返ることも可能である．家毎など自分が振り返りたい情報について検索をかけて見ることができるサービス．

スマートシティサービスのために利用されるデータは，都市内の大量のモノやシステムから収集される．したがって，その量（volume），種類（variety）は非常に大規模となる．また，状況を把握するために，なるべく最近のデータを反映する必要があり，情報の鮮度・スピード（velocity）も重要になる．これらから，スマートシティサービスのデータは**ビッグデータ**となる．

街や住居からデータを収集して分析し，活用するサービスやアプリケーションは従来から存在する．しかしそれらのほとんどは，データ容量の制限から，アプリケーションに必要とされるデータのみを必要な粒度で蓄積し，利用する形態がほとんどである．一方，クラウド時代では，データ容量の制限は事実上なくなる．よって，**生ビッグデータ（1 次データ）**を資産として蓄積し，様々な用途に横断的に活用，再利用できるようなプラットフォームが求められている．

2.2 先行研究：Scallop4SC

我々は先行研究において，スマートシティ内で取得される多種多様かつ大規模なログデータを蓄積・活用するためのスマートシティ向けデータ処理プラットフォーム **Scallop4SC** の開発を行なっている [3]．Scallop4SC のアーキテクチャを図 1 に示す．各スマートハウスからロガーを通して収集されたログ情報はネットワークを通じて，Scallop4SC の管理する分散データベース HBase 上に蓄積され，分散処理基盤 Hadoop により処理が施される．スマートシティ全体の静的な構成情報は，関係データベース MySQL 上で管理される．

さらに，Scallop4SC はアプリケーションが必要なデータに高速にアクセスし，効率的に取得できる仕組みとしてデータベースの**体現ビュー**の考え方を導入した **MVaaS（Materialized View as a Service）**を提供する．体現ビューはクエリの結果をあらかじめ実際のデータテーブルにキャッシュし，高速なデータアクセスを可能とする技術である．MVaaS の基本的なアイデアとしては，まずアプリケーションから渡される**データ仕様書（Data Spec.）**に基づいて，1 次データから体現ビューが生成される．アプリケーションは MVaaS の提供する API を介して，生成された体現ビューを利用することができる．

先行研究 [4] では MVaaS のコンセプトの提案に止まっており，具体的なアーキテクチャの検討と実際の開発は残る課題とされていた．本研究では Scallop4SC 内での MVaaS の実現を目的として，MVaaS のアーキテクチャに関する検討とその開発に取り組む．

2.3 Scallop4SC に蓄積された 1 次データ

現在の Scallop4SC に蓄積されているスマートシティの 1 次データについて説明する．Scallop4SC は，様々な種類の大規模ログを収容するために，厳密なデータスキーマを用意せずに，各ログデータをシンプルなキーとバリューで表現する．これらを分散データベース HBase に格納する．

表 1 に，我々の研究室で取得した消費電力ログの格納例を示す．各行キーは，ログがとられた日付（date）と時刻（time），

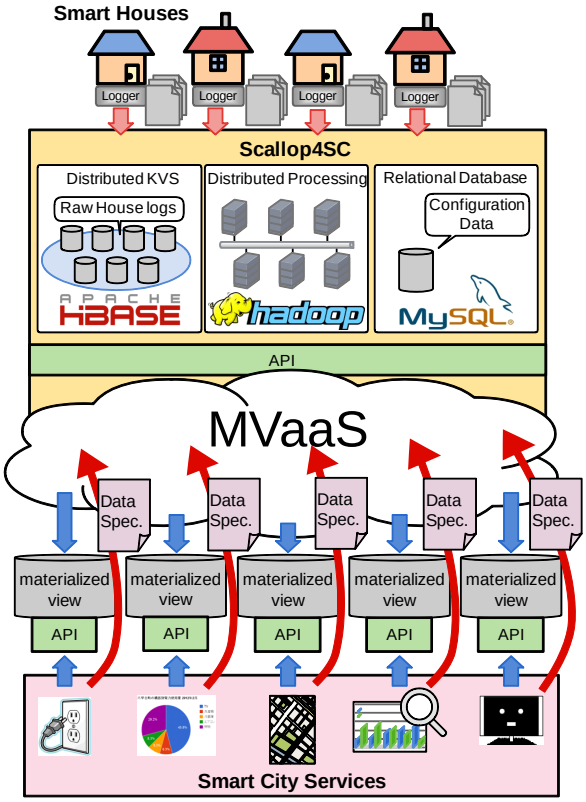

図 1　Scallop4SC のアーキテクチャ

ログの種類（type），ログがとられた住居・場所（home），ログを取得した機器（device）の連接で表現される．

各行に対して，列ファミリーは，各データを説明する属性値を示すメタデータ（info）と，データ値（data）を保持している．例えば，表の 1 行目のエントリは，2013 年 1 月 18 日 22 時 00 分 10 秒の cs27 研究室のエナジータイプ，機器 ID が pow001 の機器の瞬間消費電力値を示している．また，データの単位（unit）や，位置（location）などの付加的な情報も格納されている．

3.　MVaaS（Materialized View as a Service）

3.1　アーキテクチャ

我々は Scallop4SC 内において 1 次データから体現ビューを生成する方法として MVaaS を提案している．

図 2 に，MVaaS のアーキテクチャ図を示す．

（1） 初めに，アプリケーション開発者はどのような体現ビューが欲しいのかをデータの仕様として，MVaaS が用意している Data Spec. 入力画面の入力フォームに入力する．

（2） **MVaaS Constructor** は入力フォームに入力された情報を読み取り**データ仕様書（Data Spec.）**を生成して **MapReduce Generator** に渡し実行する．同時に **MVaaS Management DB（MySQL）**に Data Spec. の情報と体現ビューの生成タスクの優先順位を保管し管理する．

（3） MapReduce Generator が実行されると，受け取った Data Spec. に基づいて，アプリケーションが要求する体現ビューを生成するための MapReduce Program を生成する．

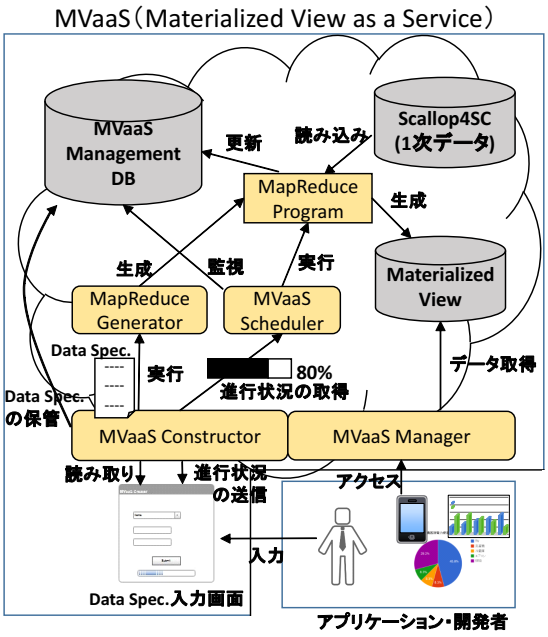

図 2　MVaaS のアーキテクチャ

（4） **MVaaS Scheduler** は MVaaS Management DB を監視しておき，優先順位の高いタスクからそのタスクの MapReduce Program を実行する．MapReduce Program は Hadoop 環境上で並列分散処理され，Scallop4SC に蓄積・保管されている 1 次データから体現ビューを生成する．なお，MapReduce Program が実行されている間，タスクの進捗状況は MVaaS Management DB に順次保管し更新する．タスクの進捗状況は DB を監視している MVaaS Scheduler によって MVaaS Constructor を通して Data Spec. 入力画面に表示される．アプリケーション開発者は常にタスクの進捗状況を確認することができる．

（5） アプリケーションは，**MVaaS Manager** の API を介して生成された体現ビューにアクセスすることで，高速に必要なデータを取得することができる．

3.2　1 次データから体現ビューへの変換

MVaaS 実装の具体的な説明に入る前に，1 次データから体現ビューを生成する手法について説明する．体現ビューの生成は MapReduce プログラムを用いた並列分散処理によって行う．

MapReduce は，巨大なデータセットを持つ並列可能な問題に対し，複数の計算ノードを用いて並列処理させるためのフレームワークである．入力される各データを指定するキーとバリューの組に変換する map 処理と，同じキーを持つ複数のバリューを集約する reduce 処理から構成される．

体現ビューの生成は以下の 3 ステップで行われる．

（1） **Filter Step**：1 次データにフィルターをかけて必要なデータだけを抜き出す．Filter 条件は $filter = [prop\ cmp\ val]$ で表される．$prop$ は表 1 の 1 次データの Column Families で指定される．cmp は $prop$ と val を比較するもので eq（==），neq（!=），gt（>），ge（>=），lt（<），le（<=）のいずれかで指定される．val は $prop$ に対する値である．

また Filter 条件を 2 つ以上用いる場合は，条件を接続して 1

表 1　1 次データの形式

Row Key	Column Families							
	info:							data:
(dateTtime.type.home.device)	date	time	device	home	location	type	unit	
2013-01-18T22:00:10.Energy.cs27.pow001	2013-01-18	22:00:10	pow001	cs27	s101	Energy	W	140.3
2013-01-18T22:00:10.Energy.cs27.pow002	2013-01-18	22:00:10	pow002	cs27	s101	Energy	W	0
2013-01-19T12:30:00.Energy.cs27.pow001	2013-01-19	12:30:00	pow001	cs27	s101	Energy	W	120.7

つの Filter 条件として表す．アンド接続 [$filter_1$ && $filter_2$] もしくはオア接続 [$filter_1$ || $filter_2$] のどちらかで接続する．

例えば「機器 ID が “pow001” で値が 20W 以上のデータ」をフィルターする場合，$filter$ = `[info:device eq pow001 && info:unit eq W && data:  ge 20]` となる．

（2） **Grouping Step**：フィルターされ抜き出された各レコード（DB の 1 行）に対して，MapReduce の map 処理においてグルーピング作業を行う．グルーピングのプロパティ [$prop_1$ $prop_2$ ... $prop_n$] を基にキーを生成しグルーピングを行う．$prop$ は 1 次データの Column Families で指定される．ただし，$date$ は $year$，$month$，$date$ に，$time$ は $hour$，$minute$，$second$ に詳細にプロパティを分けて指定することができる．例えば `[device minute]` ならキーの構成は device.year-month-dateThour-minute となる．

（3） **Aggregating Step**：グルーピングされたキーバリューの組について，プロパティ[$aggr$（$expression$)] で指定された集約方法で reduce 処理を行う．

4. 実　　装

4.1 概　　要

本研究では，MVaaS においてアプリケーションから渡された Data Spec. を解析し，体現ビューを構築する機構の実装を行う．

具体的には，MapReduce Generator が受け取った Data Spec. を解析し，アプリケーションが要求する体現ビューを生成するための MapReduce Program を自動的に生成する．

4.2 Data Spec.

MVaaS Constructor から受け取る Data Spec. は XML 形式で表現される．図 3 に Data Spec. の一例を示す．

Filter 条件

$< filter >$ の階層が 1 つの Filter 条件を表す．$< column >$ は column，$< qualifier >$ は qualifier，$< value >$ は値，$< measure >$ は column : qualifier と値の比較方法，$< label >$ はラベルを示す．$< measure >$ は 3.1 節で述べたように eq, neq，gt，ge，lt，le のいずれかで記述する．ここで例として，図 3 の $label_A$ の filter 条件を説明する．これは $info : unit$ が W と等しいものをフィルターするための条件で条件式は $filter$ = `[info:unit eq W]` となる．

Filter 条件の接続

$< connects >$ の階層で Filter 条件同士の接続方法を指定する．$< connect >$ でアンド接続かオア接続かを選択し，$< field1 >$ と $< field2 >$ で接続する Filter 条件のラベルを指定，$< label >$ は接続した後の Filter 条件のラベルを示す．ここで例として，図 3 の Filter 条件の接続を説明する．これはラベル $label_A$ とラベル $label_B$ の二つの Filter 条件を and で接続している．つまり，$label_A$ と $label_B$ の両方の条件を満たすようにフィルターすることを表していて，接続した後の条件式は $filter$ = `[info:unit eq W && info:type eq Energy]` となる．

```
<data>
  <filters>
    <filter>
      <family>info</family>
      <qualifier>unit</qualifier>
      <value>W</value>
      <measure>eq</measure>
      <label>label_A</label>
    </filter>
    <filter>
      <family>info</family>
      <qualifier>type</qualifier>
      <value>Energy</value>
      <measure>eq</measure>
      <label>label_B</label>
    </filter>
  </filters>
  <connects>
    <connect>and</connect>
    <field1>label_A</field1>
    <field2>label_B</field2>
    <label>LABEL_X</label>
  </connects>
  <info>
    <description>Sum of Data</description>
    <name>SumData</name>
    <time>2013-01-01T00:00:00,2013-01-31T23:59:59</time>
    <grouping>device,house</grouping>
    <aggregating>SUM,Data</aggregating>
  </info>
</data>
```

Filter条件

Filter条件の接続

メタ情報

図 3　Data Spec. の一例

メ タ 情 報

$< info >$ はその他の体現ビューを構成するためのメタ情報を表す．$< description >$ は体現ビューの説明，$< name >$ は体現ビューをデータベース上に作るうえでのデータテーブルの名前を表す．$< time >$ は 1 次データのスキャン範囲を date，time で指定している．図 3 の例の場合は，2013 年 1 月 1 日 00:00:00 から 2013 年 1 月 31 日 23:59:59 までのデータをスキャンすることを指定している．$< grouping >$ は 3.2 節で記述した Grouping Step でのプロパティを指定している．例の場合は，$device$ と $house$ でキーを構成する．$< aggregating >$ は Aggregating Step でのプロパティを指定している．例では Reducer での集約方法として合計 (SUM) を指定している．

この Data Spec. を MapReduce Generator が解析することで，アプリケーションが望む体現ビューを生成するための MapReduce Program が生成される．

4.3 MapReduce Generator と MapReduce Program

本節では，MapReduce Generator と Generator により自動生成される MapReduce Program について説明する．

4.3.1 MapReduce Generator

MapReduce Generator は Data Spec. に基づいて，Mapper プログラムと Reducer プログラムを自動生成する Java プログラムである．MapReduce Generator は，まずアプリケーションから渡された Data Spec. を XMLParser により解析する．Data Spec. の $< filters >$ と $< connects >$ に該当する部分を解釈し，データ選択のための Scan オブジェクトを生成する．次に，Mapper の汎用的なコード（Mapper テンプレート）をベースとして，$< grouping >$ の部分を反映した Mapper プログラムを生成する．最後に，$< aggregating >$ の部分を解釈して，集約方法を選択し Reducer プログラムを出力する．

4.3.2 Mapper

Mapper プログラムは MapReduce Generator によって動的に生成される．MVaaS における Mapper の Mapper テンプレートを以下に示す．

```
class Mapper
  method map(rowkey, result)
    for (property in grouping)
      key = concat(key, result.property.getValue())
    value = eval(expression(result))
    EMIT(key, value)
```

Mapper は 1 次データからフィルタリングされ抜き出されたデータを 1 行ずつ受け取る．その 1 行ずつに対して $(key, value)$ の組を生成する．key は $< grouping >$ で指定されたプロパティを連接したもので，連接には “.” を用いている．$value$ はデータの値（1 次データの data）とする．Mapper は最後に $(key, value)$ の組を出力し Reducer に渡す．

4.3.3 Reducer

Reducer はあらかじめ MVaaS 内に数種類用意しておき，アプリケーション開発者が望む集約方法の Reducer を選択する．現 MVaaS では 6 つの Reducer が用意されている．

- **SumReducer**：mapper から受け取ったキーバリューの組に対して，キー毎に足し算を行い結果を出力する．
- **AverageReducer**：mapper から受け取ったキーバリューの組に対して，キー毎にその平均を計算し結果を出力する．
- **MaxReducer**：mapper から受け取ったキーバリューの組に対して，キー毎にその中で値が最大のものを選択し出力する．
- **MinReducer**：mapper から受け取ったキーバリューの組に対して，キー毎にその中で値が最小のものを選択し出力する．
- **ConcatReducer**：mapper から受け取ったキーバリューの組に対して，キー毎に値を連接したものを出力する．連接に用いる文字列は Data Spec. の Aggregating プロパティで指定できる．
- **IdReducer**：mapper から受け取ったキーバリューの組をそのまま出力する．

Reducer の Reducer テンプレートを以下に示す．

```
class Reducer
  method reduce(key, [v1, v2, ..., vn])
    result = aggr(v1, v2,...,vn)
    EMIT(key, result)
```

Mapper から $(key, value)$ の組を受け取り，同じ key を持つ複数の $value$ に対して集約を行う．プログラムコードの関数 $aggr()$ の演算の部分が Reducer の種類によって異なる．例えば SumReducer なら値の合計を計算し，AverageReducer なら値の平均値を計算する．演算を行った結果を $result$ として，$(key, result)$ の組を出力し体現ビューに書き込む．

5. 評 価 実 験

5.1 実 験 概 要

MVaaS の評価実験として以下の 2 つの項目についての実験を行った．

評価実験 E1: MVaaS の効率を確かめるため，Data Spec. を受け取ってから体現ビューが生成されるまでにかかる時間を計測する．

評価実験 E2: 体現ビュー自動生成による開発コストの削減効果を確認するために，Data Spec. の行数と，その Data Spec. と等価な処理を行う MapReduce プログラムの行数を比較する．

5.2 実 験 環 境

実験で利用する 1 次データは，我々の研究グループで開発しているスマートハウス環境 CS27-HNS において，32 種類の機器から 3 秒毎に記録した消費電力ログである．計測対象の家電には，テレビや照明，エアコンなどの家電が含まれる．消費電力ログは，研究室で運用している Linux クラスタ（Pentium4, 3.0GHz, 2GB × 9 台）の上で実行されている分散データベース HBase に格納されている．自動生成される MapReduce Program も同様にこの Linux クラスタ上に設置された Hadoop 環境下で実行される．MapReduce Program は Java 言語で，Data Spec. ファイルは図 3 で示した XML の形式で記述される．なお，MapReduce 実行に使用したライブラリは，hadoop-core-1.0.3.jar, hbase-0.94.2.jar である．

5.3 評価実験 E1

5.3.1 実 験 方 法

評価実験 E1 では以下 3 つの Data Spec. を用意し，それぞれの Data Spec. に対して体現ビュー生成に必要となる時間を計測する．

A：アプリケーションの要求「2013 年 1 月 1 日から 1ヶ月間の，1 日ごとの総消費電力を取得する」に対する Data Spec.

```
filter = [info:unit eq W && info:type eq Energy]
grouping prop = [date]
aggregation prop = [SUM,Data]
time range = [2013-01-01,2013-01-31]
```

B：アプリケーションの要求「2013 年 1 月 1 日から 1ヶ月間の，機器別の 1 時間ごとの平均消費電力を取得する」に対する Data Spec.

```
filter = [info:unit eq W && info:type eq Energy]
grouping prop = [device, hour]
aggregation prop = [AVE,Data]
time range = [2013-01-01,2013-01-31]
```

C：アプリケーションの要求「2013 年 1 月 1 日から 2ヶ月間の，1 日ごとの総消費電力を取得する」に対する Data Spec.

```
filter = [info:unit eq W && info:type eq Energy]
grouping prop = [date]
```

表 2　実験結果 E1 : 体現ビュー生成にかかる時間

Data Spec.	生成時間
A	77 分 29 秒
B	62 分 13 秒
C	151 分 43 秒

```
3 aggregation prop = [SUM,Data]
4 time range = [2013-01-01,2013-02-31]
```

全ての Data Spec. は同一のフィルター条件を持つ．また A と B はスキャン範囲（1ヶ月間）が同じで，グルーピングのプロパティ（日付あるいは機器・時間）と Reducer での集約方法（合計あるいは平均）が異なる．C は A とグルーピングのプロパティと Reducer での集約方法が同じで，スキャン範囲のみを 2 倍の 2ヶ月間に設定している．

5.3.2　結果と考察

表 2 に実験の結果を示す．Data Spec. の列は前節で説明した A〜C の 3 種類の Data Spec. を意味する．

A と B の比較（77 分 : 62 分），及び A と C の比較（77 分 : 151 分）から，体現ビュー生成にかかる時間は 1 次データのスキャン範囲に大きく依存していることが読み取れる．

A と B はスキャン範囲が同じであるにも関わらず，B の方が A よりも 15 分ほど生成時間が短い．これは，生成されるキーの数と Reducer での並列効率が理由であると考えられる．グルーピングのプロパティが異なれば生成されるキーの数が異なる．A は日付をキーとする一方で，B はデバイスと時間の組をキーとするため，B の方がキーの数が多くなる．そのため，キーの数が多い B の方が Reducer の並列効率が上がり，生成にかかる時間が短くなったと考えられる．

次に MVaaS の実現可能性について考察する．スキャン範囲を 1ヶ月とした場合の体現ビューの生成には約 1 時間必要である．住宅ログは変更のないログデータであり，一度生成された体現ビューは以降の再生成や更新の必要がない．スキャン範囲が 1ヶ月の体現ビューであれば，1ヶ月に一度，1 時間の生成時間を必要とするだけである．この体現ビューの生成には定期的なバッチ処理で実現可能である．また，より長い生成時間を要する複雑な体現ビューの生成であっても，Hadoop クラスタのノード数の増加といった対処方法も比較的容易である．これらの結果から，MVaaS の実現可能性が確かめられたと考える．

5.4　評価実験 E2

5.4.1　実 験 方 法

体現ビューの自動生成による開発コストの削減効果を確認するために，Data Spec. の行数と，その Data Spec. と等価な MapReduce を開発者自身が実装する際のコード行数を比較する．この実験では 5.3 節で用いた A（1ヶ月間の 1 日ごとの総消費電力の計算）の Data Spec. を用いる．

5.4.2　結果と考察

表 3 に比較結果を示す．表より Data Spec. を記述するほうが，記述量が大幅に減っていることが分かる．このことから MVaaS を利用することでアプリケーション開発者の体現ビュー開発コストを削減できることが確認された．またプログラムを開発する場合は自分でコードを考え，バグ修正などをしなければならない．一方，MVaaS に体現ビューの生成を依頼する場合は既定の書式に合わせて必要事項を記入するだけでよいので，アプリケーション開発者にかかる開発コストを削減するのみならず，MapReduce Program の信頼性も向上すると考えられる．

表 3　実験結果 E2 : 行数の比較

	行数
Data Spec.	58
java プログラム	352

提案手法の限界として，4.3.3 節で述べた合計や平均などの 6 種類の集約処理以外の処理や，複数の MapReduce を連続して実行するような多段 MapReduce といった，MapReduce Generator の能力を超えるような MapReduce は実現できない点が挙げられる．ただし，集約処理については Reducer の部分に新たな集約関数を定義することで容易に拡張可能である．多段 MapReduce に関しては Data Spec. の表現と合わせて検討する必要があり，今後検討していく予定である．

6.　お わ り に

本稿では，MVaaS のアーキテクチャに関する検討とその開発に取り組んだ．アプリケーションから渡された Data Spec. を解析することで MapReduce Program を自動生成し動的に体現ビューを生成する．アプリケーション開発者は MVaaS を用いることで，自身で体現ビューを生成するためのプログラムを記述する必要がなくなり，MVaaS にデータの仕様を渡すだけで欲しい体現ビューを生成することができるようになった．今後の課題として，Data Spec. を生成する MVaaS Constructor，アプリケーションが体現ビューにアクセスするための API を持つ MVaaS Manager の実装を行いたい．

謝辞　この研究の一部は，科学技術研究費（基盤研究 C 24500079, 基盤研究 B 23300009），および，積水ハウスの研究助成を受けて行われている．

文　　献

[1] R.G. Hollands, “Will the real smart city please stand up?,” City: analysis of urban trends, culture, theory, policy, action, vol.12, no.3, pp.303–320, 2008.
[2] A. Mahizhnan, “Smart cities: The singapore case,” Cities, vol.16, pp.13–18, 1999.
[3] S. Yamamoto, S. Matumoto, and M. Nakamura, “Using cloud technologies for large-scale house data in smart city,” In International Conference on Cloud Computing Technology and Science (CloudCom2012), pp.141–148, Dec. 2012.
[4] S. Yamamoto, S. Matsumoto, S. Saiki, and M. Nakamura, “Materialized view as a service for large-scale houselog in smart city,” International Conference on Cloud Computing Technology and Science (CloudCom2013), (to appear).
[5] City of Yokohama, “Yokohama smart city project,” http://www.city.yokohama.lg.jp/ondan/english/.
[6] 北岡賢人，瀬戸英晴，まつ本真佑，中村匡秀，“ホームネットワークシステムにおける機器状態ログからのエネルギー浪費行動の検出,” 電子情報通信学会技術研究報告，第 110 巻，pp.37–42, 2011.