Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

クラウドを利用した電力可視化システムの構築

高野 了成[†1]　中田 秀基[†1]
清水 敏行[†1]　工藤 知宏[†1]

きめ細やかな節電計画の立案には、事業所内の分電盤ごと、さらにはブレーカごとの細分化した単位でリアルタイムに使用電力を可視化する必要がある。しかし、必要な電力計測点が多くなるため、システム全体のコスト削減やスケーラビリティの向上が重大な問題になる。我々は、安価な電力計測ユニット（計測点あたり 2500 円）とクラウドコンピューティング技術を用いることで、安価かつスケーラブルなシステムを短期間に開発することに成功した。そして、産総研のサーバ室および実験室に 290 計測点を有するシステムを導入し、個別機器毎の電力使用量を可視化したので報告する。

Developing an electrical power visualization system ulilizing cloud computing

Ryousei Takano,[†1] Hidemoto Nakada,[†1]
Toshiyuki Shimizu[†1] and Tomohiro Kudoh[†1]

Visualization of electricity consumption per segmented unit, such as a power distribution board and a breaker, in real-time is becoming essential to planning finer electricity savings. For collecting and processing a large amount of distributed measured data, the total cost and scalability of a system have become key issues. Our newly developed system helps reduce total system cost by utilizing cloud computing for data collection, and employing low-cost power measuring units (2500 JPY per measurement point). A system with 290 measurement points covering a server room and a laboratory has been successfully installed at AIST to provide a model for visualizing electricity consumption for individual equipments.

†1 産業技術総合研究所 情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)

1. はじめに

2011 年 3 月の東日本大震災後の電力不足に対応するため、電力使用制限令が発動されたことは記憶に新しいが、昨今の節電対策の要請に対して、いつどこでどれだけの電力を使用しているかを把握することは節電計画の立案には必要不可欠である。このような背景から、グリーン東大 ICT プロジェクト[1),2)] など、事業所ごと工場ごとの電力可視化システムの研究開発および普及が始まっている。よりきめ細やかな節電計画を立てるためには、事業所内の分電盤ごと、さらにはブレーカごとの細分化した単位でリアルタイムに使用電力を可視化する必要があるが、必要な電力計測器数が多くなるため、システム全体のコストやスケーラビリティが問題になる。

我々は震災後の同年 4 月から電力可視化システムの開発を始めた[3)]。開発にあたり、次の 4 点を基本方針とした。（1）電力不足が本格化する夏までの短い期間で迅速に立ち上げる、（2）既存の技術を活用し確実に動作するシステムとする、（3）安価な市販電子部品を用いて、できるだけ安価な電力計測ユニットとする、（4）測定点が後からでも容易に追加できるスケーラブルな構成とする。これらの基本方針に基づきシステムの全体構成を決定し、システムの各要素の開発を進めた。そして、低コストながら電力可視化システムとして十分な精度が得られる電力計測ユニット、およびこの電力計測ユニットの計測値をクラウド上のサーバに収集、蓄積し、様々なアプリケーションから使用電力を可視化できるシステムを開発した。本システムを用いて、産総研の計算サーバ室や実験室内 290 点を計測し、利用者がリアルタイムに計算サーバごとの使用電力を可視化できるシステムを 3ヶ月という短期間で構築できた。

以下、2 節で今回開発した電力可視化システムの全体構成と、各構成要素のハードウェアとソフトウェアの概要について述べる。3 節ではクラウドサービスの実現に用いた REST プロトコル、およびその実装について述べる。そして、システム構築よって得られた知見について 4 節で議論を行い、最後に 5 節でまとめを行う。

2. 産総研・電力可視化システム

2.1 全体構成

電力可視化システムの全体構成を**図 1** に示す。本システムは、（1）電力計測ユニット、（2）データ収集ユニット、（3）データ収集サーバ、（4）可視化アプリケーションから構成される。電力計測ユニットの電力値をクラウド上のデータ収集サーバに直接送信すると、

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

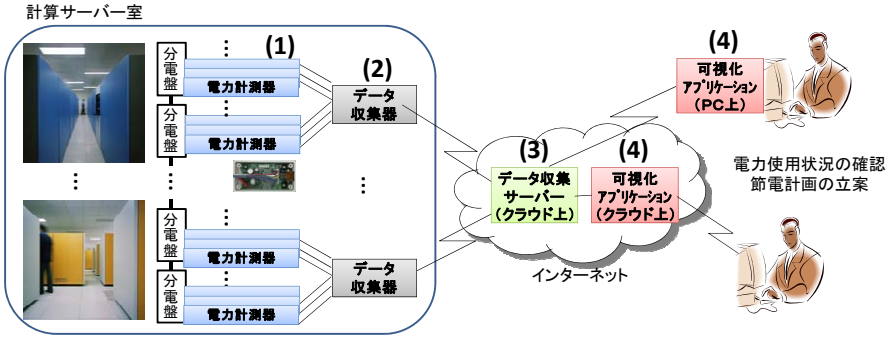

図 1 産総研・電力可視化システムの全体構成
Fig. 1 Overview of the AIST electrical power visualization system.

通信回数や通信量が増加してしまうので、データ収集ユニットで電力値を中間的に収集し、一定時間後にデータ収集サーバにまとめて送信する構成とした。また、クラウドへのアクセスなど高度な機能は電力計測ユニットから分離し、データ収集サーバに集約することで、コスト削減も狙った。

2.2 電力計測ユニット

図 2（左、中央）に示す、電力計測ユニットは電力計測基板と最大 4 つまで接続可能な電流センサから構成され、1 台で最大 4 点の同時測定が可能である。電力計測基板での信号処理にはワンチップマイコン（dsPIC30F3013）、電流センサには市販のクランプ型カレントトランスを用いることで、量産時で 1 万円程度（計測点あたり 2500 円）と安価で制作できる。本ユニットでは、単相 100/200 V であれば最大 4 点計測できる。また工場内の大型装置などでよく使用される三相 200 V についても、2 つのカレントトランスを組み合わせることで最大 2 点計測できる。以下、この 1 つまたは 2 つのカレントトランスを組み合わせた計測点のことを「プローブ」と呼ぶ。電流値と電圧値は 1 秒間に 6400 回（50 Hz の場合）または 7680 回（60 Hz の場合）サンプリングする。これにより、電流波形が高調波により複雑に乱れていても正確な計測が可能となる。

電力可視化システムで得られたデータを、課金に用いたり生産のコスト計算に組み込んだりする事例も想定されるが、その際には電力計測器の計測値の精度をいかに保証するかが課題となる。これに対して、国家標準に基づいた校正システムを開発し、10 台を抜き取り評価した結果、計測誤差は 1%程度と電力可視化システムとして十分な精度を保証できることを示した[4]。なお、電力メータとして一般的に求められる性能は 2%以下である。

また、図 2（右）のような、電力計測基板を電源タップに埋め込んだスマートタップ型センサも開発した。これによりパソコンやプリンタなど、電源コンセントにつながる個別機器毎に消費電力を測定できる。また、これとは別に河西らは Bluetooth 経由で PC から電力を可視化できるスマートタップも開発してる[5]。

2.3 データ収集ユニット

データ収集ユニットには、最大 32 台の電力計測ユニットが接続でき、各電力測定ユニットから 1 秒ごとに送信されてくる電力値のデータを収集する。収集したデータから電力値を計算、蓄積し、20 秒ごとにクラウド上のデータ収集サーバにまとめて送信する。

図 3 にデータ収集ユニットの外観を示す。データ収集ユニットは電力計測ユニットとの接続用に 32 ポート、インターネット接続およびメンテナンス用に 1 ポートの RJ-45 ソケットを持つ。電力計測ユニットとの接続には汎用の UTP ケーブルを用いるが、Ethernet とは配線が異なり、データをシリアル通信によって送信し、未使用の信号線を利用して電力計測ユニットへの電源供給も行う。このため、電力計測ユニット設置時に別途コンセントなどから電源供給回路を設置する必要がなく、工期短縮およびコスト削減が可能となった。

データ収集ユニットは 32 個の電力計測ユニット接続用ポート、CPU ボード、シリアル・パラレル変換ボード、電源部等から構成される。なお、電力計測ユニット接続用ポートに、PC 等、電力計測ユニット以外を誤接続した場合に備えた保護回路も有す。

データ収集ユニットのソフトウェアとしては、定期的に電力計測値を電力収集サーバに送信する以外に、次に示す要件を満たすことが求められる。つまり、DHCP 対応、NTP 対応、メンテナンス用の SSH アクセス、リモートからのソフトウェア更新（カーネル、ユーザランド、アプリケーション）である。そこで、安価に手に入る SH3 ボードと Linux を使用した。

具体的に、CPU ボードにはルネサス SH3 (SH7706 133MHz) を搭載した TAC 社製 T-SH7706LSR[6] を使用した。32 MB の SDRAM メモリと SDHC および 100BASE-TX Ethernet コントローラ、汎用 I/O ポート等を有する。OS やアプリケーションの格納には SDHC メモリカードを用いた。OS として SH3/Linux を用い、シリアル・パラレル変換ボードから出力されたデータを汎用 I/O ポートを介して受信するためのデバイスドライバを実装した。デバイスドライバから読み込んだデータを電力値に変換し、データ収集サーバにアップロードする処理は、Python スクリプト（pmon.py）として実装した。PC 上でのクロス開発環境および SH3 ボード上での実行環境には、組込み Linux システム構築ツールで

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

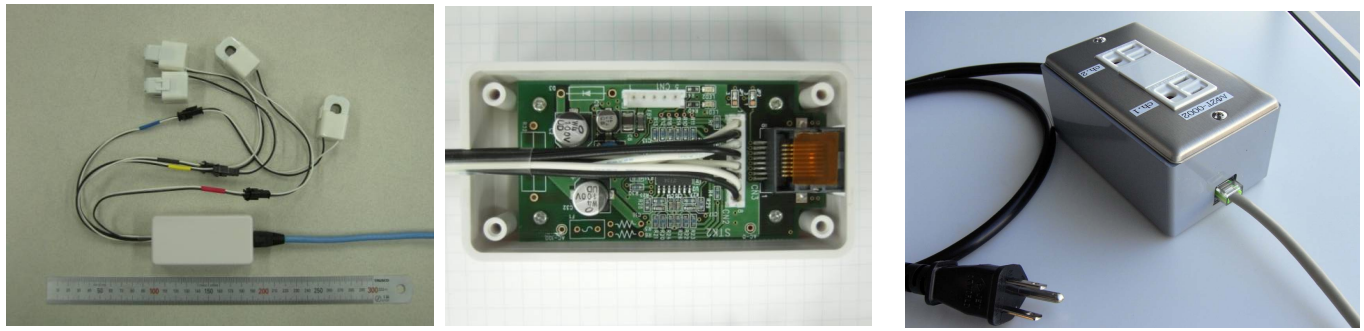

図 2 電力測定ユニット [サイズ: W 90mm x D 45mm x H 25mm] (左、中央)、電源タップ内蔵型センサ (右)
Fig. 2 Power measuring unit [size: W 90mm x D 45mm x H 25mm] (left, middle), Sensor built-in Power Strip(right).

ある buildroot[7] を使用した。作成した uClibc と BusyBox ベースのルートファイルシステムは、Python 等実行に必要なパッケージをすべて含めて 24 MB 弱と、通常の Linux ディストリビューションと比較して非常に小さい。

シリアル・パラレル変換ボードは、最大 32 個の電力計測ユニットから受信したシリアル信号を、8 ビットのデータに、カレントトランスの区別に用いる 5 ビットの識別子を加えたパラレル信号に変換して CPU ボードに渡す。当初は、このシリアル・パラレル変換も CPU ボード上で実装することを計画したが、詳細を 4.1 節に後述するように、信号処理のタイミング制約を満たせないので、FPGA に処理を分割した。実装にはゲートサイズが 250K の Xilinx Spartan-3E を用いた。

2.4 データ収集サーバ

データ収集サーバは、データ収集ユニットから送信されてくる電力値を最終的に集約し、すべての測定点の使用電力量をデータベース化する。さらに、可視化アプリケーション等に対して、任意の計測点と時間の使用電力量を取得するためのインタフェースを提供する。この通信プロトコルとして、平易な REST プロトコルを定義し、Windows や Mac OS X 等のデスクトップアプリケーションから Web アプリケーションなどさまざまなアプリケーションを接続可能とした。

実装には Google App Engine (GAE)[8] を利用した。GAE は Google 社のデータセンタインフラを利用して Web アプリケーションを実行できる PaaS クラウドサービスであり、スケーラブルかつ安定したデータストアを提供する。データ収集サーバの開発言語には Java を用い、さらに MVC フレームワークとして Slim3[9] を利用した。

図 3 データ収集ユニット [サイズ: W 400mm x D 250mm x H 60mm]
Fig. 3 Measurement data collection unit [size: W 400mm x D 250mm x H 60mm].

2.5 可視化アプリケーション

データ収集サーバ上に蓄積されたデータベースから電力使用量をダウンロードし、ユーザが視認できるようにするために、2 種類のアプリケーションを開発した。

図 4（左）に示したアプリケーションは、HTML5 で実装されており、クラスタ計算機など任意の集合の電力使用量を、1 分ごとに積み上げグラフとして表示する。一方、図 4（左）に示したアプリケーションは、Mac OS X 用のデスクトップ常駐型アプリケーションであり、興味のある機器の電力使用状況をアイコンにより通知する。

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

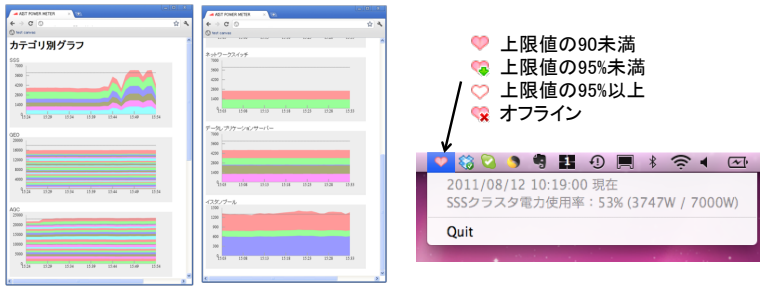

図 4 可視化アプリケーション例: ブラウザアプリケーション（左）、デスクトップアプリケーション（右）
Fig. 4 Visualization ppplications: browser application (left), desktop application (right).

3. 電力可視化 REST プロトコルの設計と実装

3.1 設　　計

クラウド上のデータ収集サーバとの通信には、平易な Representational State Transfer (REST) プロトコルと JavaScript Object Notation (JSON) データ記述言語を採用した。

図 5 に各システムコンポーネントと REST プロトコルの関係を示す。各プローブで計測された使用電力データは、データ収集ユニットから定期的にアップロードされ、データ収集サーバのデータストアに蓄積される。そして可視化アプリケーションの要求に対して、任意の地点、時間の使用電力データがダウンロードされる。プローブはデータ収集ユニット、センサ（データ計測ユニット）、プローブの階層構造を持った ID で識別される。データ収集ユニット ID はシステムグローバルで一意である必要があり、センサ ID はユニット内で一意である必要がある。現在、データ収集ユニット ID は、ユニット内の DIP スイッチで設定している。また、ユニットとセンサの組合せ、プローブの補正設定などのコンフィグ情報は、データ収集ユニットがデータ収集サーバから取得する。

表 1 に REST API の一覧を示す。データのアップロード、ダウンロード、検索、コンフィグ用の機能を提供する。URL `http://xxx.appspot.com/$path` に対して、method で示す HTTP リクエストを実行した場合、説明で示した結果が得られる。各データの授受には JSON 形式のメッセージを使用する。そのデータフォーマットを**図 6**、**図 7**、**図 8** に示す。なお、コンフィグに関しては、ユニット毎に設定を登録する代わりに、利便性を考えて、CSV 形式のコンフィグ設定を一括して登録する Web インタフェースも提供する。これより

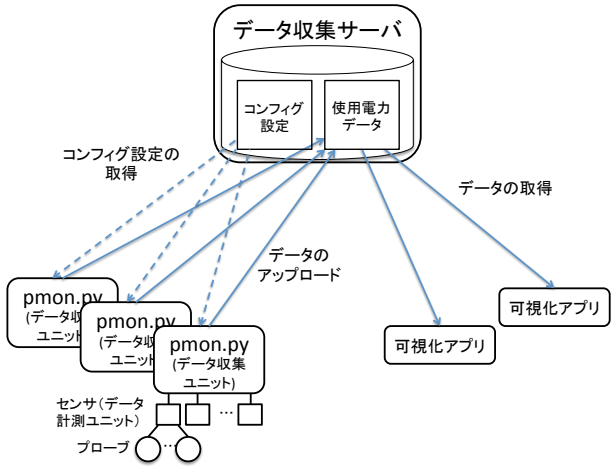

図 5 電力可視化 REST プロトコル
Fig. 5 Power visualization REST protocol.

Excel や Google Docs によるコンフィグ設定の一元管理が容易になる。

例えば、`http://xxx.appspot.com/latest` への GET リクエストに対して、次のような結果が得られる。これは 2011 年 10 月 29 日 6 時 31 分時点の各計測点の消費電力（単位はワット）を示している。なお、「#」以降は論文執筆時に挿入したコメントであり、実際の実行結果には含まれない。

```
{
    "power": {
        "aist.a02.2-12.1F.0:01141-1.p0": [1844.0], # 単相 100V x 4 プローブ
        "aist.a02.2-12.1F.0:01141-1.p1": [82.0],
        "aist.a02.2-12.1F.0:01141-1.p2": [341.0],
        "aist.a02.2-12.1F.0:01141-1.p3": [55.0],
        "aist.a02.2-12.1F.0:01141-2.p0": [831.0],  # 三相 200V x 2 プローブ
        "aist.a02.2-12.1F.0:01141-2.p1": [2779.0],
        :
    },
    "time": 1319837460,
    "timeStr": "201110290631"
}
```

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

表 1 電力可視化 REST API 一覧
Table 1 Power visualization REST API list.

path	method	説明
/update	POST	データのアップロード
/latest	GET	最後の 1 分間の平均消費電力の取得
/latest,COUNT	GET	最新 COUNT 分の平均消費電力の取得
/summary.s/YYYYmmDDHHMMSS,N	GET	指定時刻から N 秒のデータの取得
/summary.m/YYYYmmDDHHMM,N	GET	指定時刻から N 分のデータの取得
/summary.h/YYYYmmDDHH,N	GET	指定時刻から N 時間のデータの取得
/summary.d/YYYYmmDD,N	GET	指定時間から N 日のデータの取得
/query.s/LOC/YYYYmmDDHHMMSS,N	GET	指定場所 LOC の指定時刻から N 秒のデータの取得 (LOC は prefix match)
/query.m/LOC/YYYYmmDDHHMM,N	GET	指定場所 LOC の指定時刻から N 分のデータの取得 (LOC は prefix match)
/unit-config/UNIT_ID	GET	UNIT_ID のコンフィグの取得
/unit-config/UNIT_ID	PUT	UNIT_ID のコンフィグの設定

3.2 データ収集ユニットの実装

データ収集ユニットは、Python スクリプト (pmon.py) によって、プローブの電流値と電圧値を読み込んで電力値を計算し、データ収集サーバにその値を送信する。

pmon.py は起動時にデータ収集サーバからコンフィグ情報を取得し、内部のデータ構造を初期化する。コンフィグ情報には、センサ設置時の計測で調査した、各プローブの位相調整 ($PhAdj$) と倍率調整 ($MpAdj$) が含まれる。電力計測ユニットからは、1 秒毎に位相が 90° 異なる電圧 (Vcos, Vsin)、プローブごとの位相が 90° 異なる電流 (JI, JQ) が得られる。有効電力 (PWI)、無効電力 (PWQ) は次に示す計算から求めることができる。

$$Vabs = \sqrt{Vcos^2 + Vsin^2}$$
$$KI = MpAdj \times cos(\pi \times \frac{PhAdj}{180}), \qquad KQ = MpAdj \times sin(\pi \times \frac{PhAdj}{180})$$
$$PWI = \frac{Vcos}{Vabs} \times (KI \times JI + KQ \times JQ) + \frac{Vsin}{Vabs} \times (-KQ \times JI + KI \times JQ)$$
$$PWQ = \frac{Vsin}{Vabs} \times (KI \times JI + KQ \times JQ) - \frac{Vcos}{Vabs} \times (-KQ \times JI + KI \times JQ)$$

pmon.py は毎秒の有効電力を 20 秒分蓄積し、図 6 で示したアップロード用の JSON 文字列を生成し、データ収集サーバに送信する。電力量はこの有効電力を所定時間積分することで求まる。なお、無効電力は力率などを求めることに使えるが、現在は無視している。

```
{
     "id":   "データ収集ユニット ID"
     "time": "データ取得時間（UNIX epoch 時間からの経過秒数）"
     "power":  {
        "sensor0.0": [VAL0, VAL1, VAL2, ..., VAL19],
        "sensor0.1": [VAL0, VAL1, VAL2, ..., VAL19],
        "sensor1.0": [VAL0, VAL1, VAL2, ..., VAL19],
        ....
     }
}
```

power の各要素はプローブの識別名とその値のペアである。識別名はセンサー名とプローブ番号を"." で連消した文字列であり、値はプローブの time から 20 秒間における各秒の使用電力（単位はワット）を示す。

図 6 アップロード用 JSON データフォーマット
Fig. 6 Upload message formatted in JSON

3.3 データ収集サーバの実装

データ収集サーバでは、1 分ごとの秒単位のデータ (PowerMinute)、1 時間ごとの分単位のデータ (PowerHour)、1 日ごとの 1 時間単位のデータ (PowerDay) を管理している。

データ収集ユニットからのアップデートの際、データは PowerMinute のエンティティ（レコード）として格納される。それと同時に平均値を算出し、PowerHour の該当する分フィールドを更新する。この PowerHour の更新は、GAE のタスクキュー機能を用いてバックグラウンドタスクとして実行される。また、1 時間に 1 度 cron タスクが毎正時に起動し、PowerHour に収められた情報を用いて PowerDay エンティティを更新する。なお、データ収集ユニットから受信する電力データは float 型であるが、データストアへの保存量を削減するため、仮数部だけを short 型で保存し、指数部はセンサ毎に持つ。

データ収集サーバは、可視化アプリケーションからのリクエストに応じて、データストアを検索し返答する機能を持つ。各エンティティのキーに時刻と場所をエンコードしているため、特定の時刻、特定のセンサに対する問い合わせは、比較的低速なデータストアの検索機能を用いず直接読み出すことができる。

また、リクエストに対する返答をキャッシュすることで、検索機能への負荷を低減している。ブラウザ上の可視化アプリケーションは同一のリクエストをデータ収集サーバに対し

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

```
{
    "time": "データ取得時間（UNIX epoch 時間からの経過秒数）"
    "timeStr": "データ取得時間（文字列表記 YYYYmmDDTHHMM）"
    "power": {
        "LOCATION0": [計測値 LOCATION0 の電力値（W）]
        "LOCATION1": [計測値 LOCATION1 の電力値（W）]
        "LOCATION2": [計測値 LOCATION2 の電力値（W）]
        "LOCATION3": [計測値 LOCATION3 の電力値（W）]
        ...
    }
}
```

LOCATION は":"と"."で構造化された文字列。例えば、"aist.a02.2-10E.4F:411.A0"は産総研 筑波第 2、2-10E 棟、4F に設置されたボードから取得された、411 号室 A0 センサの情報を表している。

図 7 ダウンロード用 JSON データフォーマット
Fig. 7 Data download message formatted in JSON.

て発行する。このため、ブラウザ上の可視化アプリケーションがいくつ起動していたとしても、実際にデータストアの検索が行われるのは 1 分に 1 度だけである。

3.4 設置事例

産総研内の計算サーバ室 249 点および共用実験施設 41 点の合計 290 点を計測し、利用者がリアルタイムに個別機器毎の使用電力を可視化できるシステムを構築した。計測機器には計算サーバ、ネットワークスイッチ等の情報機器をはじめ、空調電源や定格 90kW の実験設備等が含まれる。

サーバ室には 4 台のデータ収集ユニットと 66 台の電力計測ユニットを設置した。その計測点数は 249 個となる。さらに共用実験施設には 3 台のデータ収集ユニットと 16 台の電力計測ユニットを設置した。その計測点数は 41 個となる。**図 9** はサーバ室内の分電盤に設置された電力計測モジュールである。各電線にクランプ型カレントトランスが取り付けられていることがわかる。**図 10** はフリーアクセスに設置されたデータ収集ユニットであり、28 台の電力計測ユニットと接続され、電力データを収集している。

データ収集ユニットの CPU ボードにおいて、CPU 使用率は電力計測ユニットを 1 台使用時のピーク値で 10%、20 台で 60%、28 台で 80%であった。これより 32 台使用時のピーク CPU 使用率は約 90%と見込まれ、最大構成時でも十分な処理性能を満たすことがわかった。

データ収集サーバは GAE 上の 1 インスタンスとして実行される。現状では、データ収集

```
{
    "id": "データ収集ユニット ID"
    "description": "当該ユニットの設置場所などの情報"
    "sensors": [
        {
            "id": "センサ ID"
            "description": "当該センサの設置場所などの情報"
            "combinations": [ct1, ct2, ct3, ct4]
            "probes": [
                {
                    "scale": 測定値の倍率
                    "description": "当該プローブの測定対象などの情報"
                }, ...
            ]
            "CTs": [
                {
                    "factors": [Fpa, Fma, Foi, Foq],
                }, ...
            ]
        }, ...
    ]
}
```

sensors は最大 32 個、probes は最大 4 個。combinations はカレントトランスとプローブの組合せを示す。例えば、単相 100V のプローブが 4 つの場合は"[0 1 2 3]"、三相 200V のプローブが 2 つの場合は"[0 0 1 1]"となる。また、入力ポートにカレントトランスが接続されていない場合は-1 と記す。カレントトランスの factors は順に位相調整、倍率調整、ゼロ点補正に用いるパラメータを示す。現在用いているカレントトランスはゼロ点補正の必要がなかったので、先頭から 2 つだけを使用している。

図 8 コンフィグ用 JSON データフォーマット
Fig. 8 Config message formatted in JSON.

ユニットからの入力によって恒常的に秒間平均 0.75 回の HTTP リクエストが行われている。受信通信帯域は平均 5KB 程度である。4.2 節でも後述するが、システム導入以降、安定した動作を続けている。

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

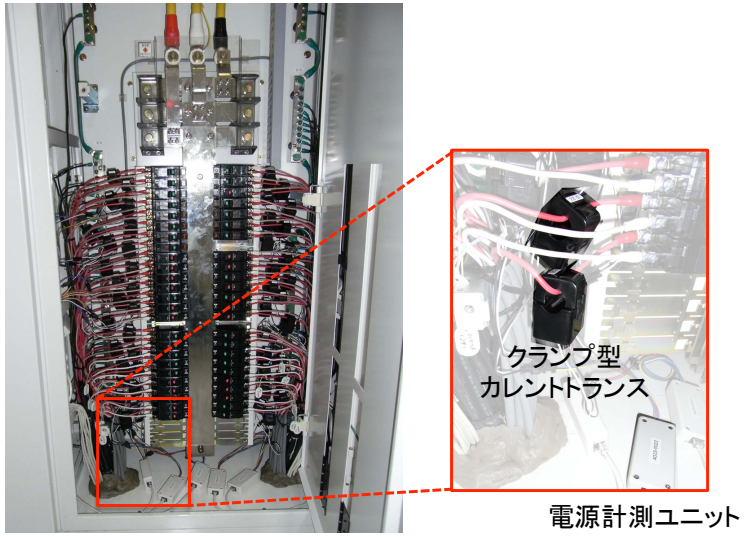

図 9 分電盤に設置された電力計測モジュール
Fig. 9 Installation of power measuring modules on a power distribution board.

図 10 フリーアクセスに設置されたデータ収集ユニット
Fig. 10 Installation of a data collection unit in free access floor.

4. 議　　論

4.1 Linux のリアルタイム性能

開発開始当初は、32 台のデータ計測ユニットからの信号を、Linux 上でデコードする予定でいた。各入力は 2400 bps のシリアル信号であり、調歩同期方式で送受信する。この信号をソフトウェアでデコードするには、ビット毎に最低でも 1/2400 秒、割込みを取りこぼすことを考慮すると、さらに 4 倍程度にオーバサンプリングした割込み周期が必要となる。つまり 1/9600 秒で約 104 マイクロ秒周期の割込み処理が必要となる。

そこでまず、タイマ管理ユニット (TMU2) 割込みを上記の処理に占有することで、104 マイクロ秒周期の割込みを発生させ、データをサンプリングした。しかし、数秒に 1 度の頻度で 200〜400 マイクロ秒区間の割込みの受付に失敗し、この結果、20 秒に 1〜2 回程度の頻度で電力値の取得に失敗した。この原因はカーネル処理内で割込禁止区間があるためと考える。割込み優先度を適切に制御することで、TMU2 割込みだけを他の処理に優先して実行することは原理的には可能と考えるが、開発時間の関係で断念した。

そこで、もう 1 枚用意した SH3 ボード上で組込み OS MES[10] を動作させ、そのアプリケーションとしてシリアル・パラレル変換を実装した。この際、割込みの代わりにポーリングを使用した。これにより電力値の取得に失敗することはなくなった。さらに、1 ユニットにつき 2 枚の SH3 ボードを使用することは高コストにもつながるので、最新バージョンでは、この処理を FPGA にて再実装した。

4.2 Google App Engine の可用性とコスト

クラウドサービスに対する懸念として、その可用性とコストが十分に要求を満たすかという点が挙げられる。

GAE のデータベースであるデータストアは分散テーブル型 NoSQL である Bigtable[11] 上に構築されており、Megastore 複製[12] によって複数のデータセンタにリアルタイムで複製される。そのため障害に対する高い可用性を有す。GAE の有料アプリ（Paid Apps）タイプのサービスレベル契約は 99.95%の可用性を謳っており、これは月あたり 1.5 日以上のサービス停止時間が発生すると、料金の 10〜50%が払い戻されることを意味する。しかし、我々が使用した直近の 4ヶ月間で GAE 側の問題によるサービス障害はなかった。また、今年の 7 月に、GAE のメンテナンスのため、複数のデータセンタが停止したが、本システムの動作には全く影響がなかった。

続いて、GAE の課金に関して簡単に議論する。GAE では CPU 使用率、ネットワーク

Vol.2011-OS-119 No.5
Vol.2011-EMB-23 No.5
2011/11/29

帯域、データストア、API 使用回数などの資源利用がクォータ方式によって制限されており、無料クォータを超える利用が課金対象となる。現在の運用では、課金の上限額を 1 日あたり最大 1 ドルに設定しているが、CPU、ネットワーク帯域などは無料クォータ内に十分に収まる程度の処理量である。一方、データストアに関しては、データ総量が無料クォータの 500 MB を超えると課金が発生するので、古いデータを削除している。この設定は運用ポリシとコストの兼ね合いで調整する必要があると考える。

以上から、GAE はサーバの維持費用がかからないことはもちろんのこと、サービスに伴う課金も十分に安いことから、コスト効率は高いと考える。ただし、年内に予定されているサービス正式化に伴う料金改正により、使い方によっては費用の増加が懸念される。それでも本システムに限れば、有料アプリとして運用しても、基本使用料の月額 9 ドル程度の課金になるものと見積もっている。

4.3 不足する機能

データ収集ユニットの死活監視は今後の課題である。GAE への定期的なアップロードをハートビートとみなすことができるので、GAE ダッシュボードのログ等を調べることで死活確認は可能である。しかし、データ収集ユニット数がさらに増加する場合、なんらかの通知機構は必須になると考える。

5. ま と め

クラウド技術を活用することで、低コストかつスケーラブルな電力可視化システムを 3ヶ月という短期間で構築できた。本論文では、産総研・電力可視化システムの概要とクラウド接続に用いた REST プロトコルの設計と実装について述べた。今回の成果は、多くの様々な機器を利用している事業所や工場における機器ごとのきめ細かな電力可視化システムの構築にも適用可能であり、大口需要家の電力使用量の削減への貢献が期待される。

今後の課題として、節電対策により効果的なアプリケーションの開発や、他サービスとの連携も含めて、取得したデータを有効に活用する応用方法の検討が挙げられる。我々はデータセンタ内の省電力化を目指し、仮想計算機マイグレーションを用いた動的サーバ集約と機器の電源断制御による省電力技術の研究開発を行っており[13]、今回開発した使用電力可視化技術の応用を検討している。具体的には、サーバ集約化のスケジューリングに各サーバの電力使用状況をリアルタイムにフィードバックすることで、さらなる集約の効率化および消費電力量の削減が期待できる。

謝辞 共同開発者であり、特に電力計測ユニットの開発を担当した、産総研情報技術研究部門スマートグリッド研究グループの村川正宏氏、河西勇二氏に感謝する。なお、本研究の成果の一部は、独立行政法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務「グリーンネットワーク・システム技術研究開発プロジェクト（グリーン IT プロジェクト）」の成果を活用している。

参 考 文 献

1) 吉田　薫，江崎　浩：グリーン東大工学部プロジェクトにおける取組みと成果，電子情報通信学会技術研究報告，Vol.109, No.351, pp.1–6 (2009).
2) 中島高英：クラウド型コンピュータによる消費エネルギーの見える化の実用事例〜グリーン東大工学部プロジェクトにおける事例紹介，電子情報通信学会技術研究報告，Vol.109, No.351, pp.7–12 (2009).
3) 産総研プレス発表資料：電力可視化システムを低コストで構築，http://www.aist.go.jp/aist_j/press_release/pr2011/pr20110902/pr20110902.html (2011).
4) 昆盛太郎，河西勇二，村川正宏，樋口哲也：消費電力の見える化とその評価・校正技術の開発，電気学会計測研究会講演予稿集 IM-11-036 (2011).
5) 河西勇二，村川正宏，樋口哲也，昆盛太郎：消費電力見える化システム〜Bluetooth 経由で PC に直接データ収集・表示〜，電波技術協会報 2011 年 11 月号 (2011).
6) T-SH7706LSR: http://www.tacinc.jp/T-SH7706/T-SH7706LSR.htm.
7) Buildroot: http://buildroot.uclibc.org/.
8) Google App Engine: http://code.google.com/appengine/.
9) Slim3: https://sites.google.com/site/slim3appengine/.
10) Micro Embeded System(MES): http://mes.sourceforge.jp/mes2/.
11) Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.E.: Bigtable: A Distributed Storage System for Structured Data, *Proceedings of the 7th Symposium on Operating System Design and Implementation (OSDI)*, pp.205–218 (2006).
12) Baker, J., Bond, C., Corbett, J., Furman, J., Khorlin, A., Larson, J., Leon, J.-M., Li, Y., Lloyd, A. and Yushprakh, V.: Megastore: Providing Scalable, Highly Available Storage for Interactive Services, *Proceedings of the 5th Conference on Innovative Data Systems Research (CIDR)*, pp.223–234 (2011).
13) Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: Reactive consolidation of virtual machines enabled by postcopy live migration, *Proceedings of the 5th international workshop on Virtualization technologies in distributed computing (VTDC)*, pp.11–18 (2011).