Actuation-Based Energy Dashboard Project with a Layered-Resolution Database

Anthony Nguyen
University of California, Los Angeles
anthony.c.nguyen@ucla.edu

Linyi Xia
University of California, Los Angeles
linyix@ucla.edu

Jeremy Haugen
University of California, Los Angeles
jhaugen@cs.ucla.edu

ABSTRACT

Recent years have seen the expansion of home energy dashboards into the marketplace. The primary focus of these dashboards has been to display information about building energy usage to the user. In this project we have built an energy dashboard capable of being a full home energy management tool. Users can see a mapping of their building, with fine grained information and control over compatible devices. Using an actuation-based approach, we can encompass the notification and information display of existing products, in addition to giving users fine-grained control over the energy usage.

The data we worked with was typically high resolution (1 Hz), and several year's worth of data was available. In order to display information that covers a large time-span, we found we needed to condense information into smaller resolutions so that it would be easy to retrieve and process for a dynamic web app. In order to accomplish this, we created a layered resolution database system, where the high resolution data was summed into lower resolution time-slices.

1. GOAL

The goal of this project is to introduce a way to structure an energy dashboard surrounded around actuation. By actuation, we mean one of the following: notification, display of information, and real-world physical manipulation. Another goal of this project was to make large amounts of data easily accessible to the application. This would allow us to plot and show statistics for energy usage from up to a year's worth of high resolution (1 Hz) data. In order to accomplish this we propose a layered resolution database model.

2. STATE OF THE ART

Current energy dashboards are typically used to monitor real-time and historical energy data for buildings. They are typically web applications, accessed using a web browser. Web-based applications are a useful medium for energy dashboards because they are multiplatform and they may allow access to multiple users. Web-applications can also be conveniently hosted on a machine which is also performing the task of data collection. This allows for a single hub to be used as a whole-home energy dashboard solution. Web applications are also useful because they can be made to only allow access to users on the local network, simplifying security concerns.

Some energy dashboards can attempt to forecast future energy usage. They may use engineering methods, statistical methods, or artificial intelligence methods to perform such forecast [1]. Energy usage forecasting is useful because it allows users to adjust their behavior in order to fit their energy usage needs more accurately. This is a desirable feature in any energy dashboard but it is not the primary focus of this project.

Another key feature of many energy dashboards is to make recommendations to the user, to produce energy savings. Occasionally, such systems are subjective to the programmer of the algorithm. However, data taken from similar buildings could be used to make a more systematic recommendation [2]. Another approach to recommender systems is to have a meta-recommender which chooses from different recommendation algorithms, using a network of users who rate the performance of each algorithm [3]. Systems have also been developed that can make recommendations to individual occupants of the building, by iteratively training on their behavioral responses to the recommendations given [4]. Making useful recommendations to users is one step toward reducing energy consumption. However, while the information is useful, it may not always be convenient for users to act on the information.

Currently, there are no commercially available databases that is suitable for our application. Under the requirement of large scale and multi level resolutions, we found some other graphic related practices. In a recent paper [6], the research group encountered the challenge of geographic data formatting and storage. Geographical data often require a large scale database, which also burdens the process of mathematical computations performs on these datasets. They employed concepts from digital image processing assuming the graphics are usually continuous where they can bring down the resolution. However, in our application, energy and water consumption are even harder to predict and more spontaneous. Also, we would hope to monitor abnormal activities over time.

The primary purpose of most energy dashboards is to educate users, allowing them to influence their energy usage habits. However, there has been a relative lack of dashboards that focus on actuation of this information in a single, seamless application. The goal of this project is to create a system that not only displays information, but implements notifications and real-world manipulation of devices with an intuitive interface.

3. NOVELTY

There appears to be a disconnect between obtaining the energy usage information and applying that information with useful effect. Often energy dashboards simply allow information to be obtained from the interface. In order for useful action to be taken, several additional steps are usually required. The ability to turn on and off devices is usually handled within a separate application. Instead, we propose a system that includes notifications and real-world actuation within a single system. We believe this increases the effectiveness of the energy dashboard.

4. IMPORTANCE

The end goal of most energy dashboards is to help users reduce their energy consumption. Making a more efficient dashboard, that includes actuation, can help users further reduce their energy consumption.

In addition, we find a solution to querying large spans of high-resolution time-series data, by using a layered resolution approach to data storage. This is an important and probably necessary technique for interactive applications utilizing high-resolution data.

5. DESCRIPTION

Our application has multiple calculations that we wish to display to the user. It is important that these calculations are useful to them, and not simply a flood of data. The particular energy usage calculations that we chose to display are:

- The water/energy usage and estimated cost since the 1st of the month
- The water/energy usage and estimated cost last month
- The water/energy usage for the current day and yesterday

- A live graph showing the whole house water/power usage for different time periods up to a year in the past
- A pie chart showing the relative power usage of individual devices
- A pie chart showing the relative water usage between different days of the week, for the past week
- A heat-map, where color indicates power usage of devices, overlaid on a map of the home

We also want to have the ability to notify users of their current estimated water/energy bill and actuate, allowing certain devices to be enabled, disabled, or changed via the web page.

5.1 Data Collection

Data is drawn from a python application that funnels the sensor data into several queues, one for each subscribing driver. We wrote a subscribing driver for this application that sends data to our data storage system (sMAP).

The data collection python application runs on an Apple Mac mini in the home, but since it is written in python, it could be ported to many other platforms.

5.2 High-Resolution Time-Series Data Storage

After the data is collected, the data must be stored in a convenient way. Because we wish to be able to access a large amount of historical data, the data storage system must be robust, and designed to handle large amounts of data. sMAP is a project out of UC Berkeley, which is open and free, and is designed for this purpose.

5.2.1 sMAP

The data is sent to a machine at another location which runs a sMAP server. We chose to send the data outside the home for storage convenience, but the sMAP server could have been located on the data collection hub as well.

The data collection driver sends the data to the sMAP server via the sMAP python library. In sMAP we must create a "data stream" for each sensor. Each data stream has a UUID, a path, and associated metadata, such as the units that the data will be in. The path is simply a hierarchical description of the sensor. For example, "/myhouse/masterbedroom/light1" might be the path for a sensor that tells whether a particular light in the master bedroom is on or not. For each sensor, we create a path, and for convenience we set the UUID to be the MD5 checksum of the path. Accessing the data from sMAP is generally easiest using the UUID. Therefore, having a standard way to generate the UUID from the path allows all our application components to be written in terms of convenient sensor names, but also access the data in the simplest way possible.

The sMAP data points must be in the form of timestamp, value pairs, which is necessary for applications using time-series data. In addition to adding real-time sensing data to sMAP, we also had access to three year's worth of sensing data for the particular home our application was monitoring. Therefore we added this data using sMAP's archival bulk data submission mechanism.

5.3 Back End

5.3.1 Django

For the back-end of our web app, we used the python web framework, Django. It simplifies the process of creating web apps, by providing the ability to write html templates. Html templates allow us to easily reuse and dynamically create html code.

Django also provides the ability to route web addresses to function calls. This allows us to call functions using a RESTful API.

5.3.2 Layered-Resolution Database

Several of the functions for the energy dashboard require us to query data for large time periods, such as 1 month or 1 year. Much of the data in sMAP is high resolution

(1 Hz sampling frequency). If we were to plot the energy usage for the past 1 year by retrieving all the data from sMAP, we would get 365*34*60*60 = 31,536,000 samples. Even if we optimized our REST API to use binary data, and reduced the resolution of the samples to 1 byte, we would still have around 31 MB of data for each one-year query. This is far too much for an interactive web app to handle. Therefore we maintain a separate relational database with data stored in smaller resolutions. For example, one of our database tables includes the sum of the energy readings for every day from the past year. This allows us to do our one-year query by only retrieving 365 entries. We also maintain a table which stores the sum of the data-points for every hour, allowing us to plot the energy usage for the past day/week without needing to retrieve too many data points.

There many additional complications related to maintaining the data in multiple databases. First, we need to keep the relational database in sync with sMAP. We do this using a simple Cron script that runs 1 minute past every hour. This Cron script querys sMAP for the data from the last hour, and updates the tables in the database. Because of this, our data is not always up-to-date in our database. Therefore, we may need to check a higher resolution database when the database we are looking at doesn't have the information we need.

Another complication is if we require our data to be current, then we must query multiple databases and combine results. For example, consider doing the calculation for a user's monthly bill. First we must query one database table, to get the usage from the first of the month through yesterday. Then we must query another database table to get the hours from midnight today through the last hour. Then we must query sMAP to get the current hour's usage. In fact it is even more complicated than this, since the alignment of the data matters. In our case, the "daily sum" table stores data for every day in terms of UTC time, not the user's local time. Therefore, the data-point for the first day of the month may not fall exactly on the boundary for the user's month. In order to resolve this issue, we must again query the "hourly sum" database table to get the usage from the users start of the month, to the first valid day in the "daily sum" database.

Figure 1 shows a simplified version of how our different databases work together. In the figure the top row represents the lowest resolution database, while the bottom row represents the highest resolution database. Each rectangle represents a data-point in the respective database. Each rectangle (data-point) is the sum of the rectangles directly below it.

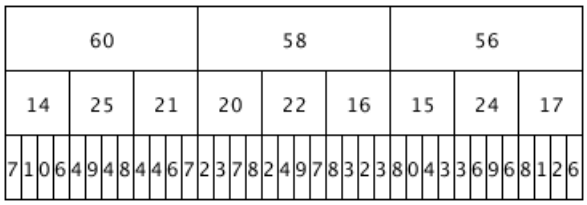

Figure 1: Layered Data Resolutions

By summing up the results of each query, we can find the total usage for a specific range of time. We can also find the average usage by finding the total usage for the time period, then finding dividing by the time.

In the general case, if a person has n levels of data granularity (in our case n = 3, for seconds, hours, and days), then to get the total usage from one arbitrary timestamp to another arbitrary timestamp, they must query a maximum of $(2n - 1)$ different time ranges. Figure 2 shows an example of how the query might be split between the different data resolutions for a particular query. The gray, rectangles are the data-points that must be queried in order to gain information about a specific range of time.

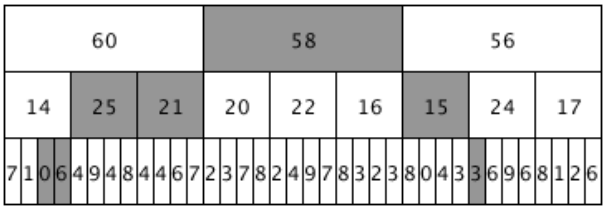

Figure 2: A Particular Query for the Layered Resolution Database Model. Queried Data-points In Gray.

In this particular example, we are retrieving the total of the readings from time-slice 3 to time-slice 29 inclusive, by only retrieving 7 data-points, instead of 29 - 3 + 1 = 27 data-points. This type of model can cause great savings in terms of data processing, at the expense of some overhead in maintenance and extra memory costs. The exact savings is difficult to quantify in the general case since it depends on the number of layers, their resolutions, the length of the time-span of the query, and the time offset of the query from the boundary of the low-resolution layers.

We believe that this type of model would be an interesting and useful feature to build into sMAP-like databases. It could potentially work similar to how indexes are added to relational databases. In a relational database, an index can be added to a column in order to speed up data retrieval and sorting at the expense of initial overhead and memory. This is a similar tradeoff to our model. Having this feature built into sMAP would greatly increase the convenience and usability of this type of model, since it could handle all the maintenance and sub-query processing on its own.

There are several types of queries where this model is useful: total usage, which simply sums up all the data-points retrieved; average usage, which sums up all the data-points retrieved and divides by the time; and windowed average/total usage. A windowed usage query is when you want to find the average or total usage for multiple time-windows. This type of query is particularly useful for plotting the usage over time. For example, when we plot the average usage over one year, we use a window size of one day, so that we plot 365 data-points. To do a windowed query, there are a few parameters that are needed: a start time, an end time, a window size, and an operation (typically *average* or *sum* in our case). A windowed query is essentially *n* average/total usage sub-queries, where *n* is the number of windows retrieved. The windowed query is therefore subject to the same benefits that the average/total queries get from our model. However, since it takes a factor of *n* more time, it can be greatly enhanced by creating optimal conditions. The optimal conditions for this type of query are when the start time and window size match exactly to one of the upper layers. This essentially means that the query is only applied to one layer of the database model, each data-point corresponds to one window, and no additional data retrieval is needed.

sMAP already provides a query system that can retrieve window queries. However, it does not use the layered resolution model that we use. Therefore, if trying to get the windowed average usage with a window size of 1 day, for a total of 1 year, with 1 Hz data resolution, sMAP would need to look at about 31 million samples. In fact, sMAP doesn't even allow queries that would require analyzing this many data-points in its built-in query system. (Although, one could retrieve the raw data themselves to do the calculation). The built-in query system only looks at a maximum of about 10,000 data-points. With the layered resolution data model, sMAP would be able to handle this type of query more easily, especially if it knew what the window size and alignment of the start time would be in advance. It could create a layer in the model optimized for that query, and maintain it as data is added or modified. This would give database users a similar experience to how indexes are added to relational databases, but with a little more involvement, since the window size and alignment would need to be specified.

5.4 Front End

The front end is supported by Django framework. After the required files

(manage.py, views.py and urls.py), the front end runs in a similar way as many other services. In a system snapshot, the HTML5 files are generated on the go as users click on different tabs and pages.

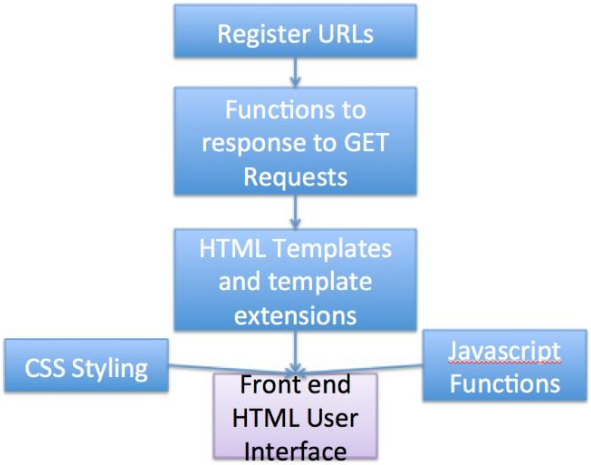

Figure 3: Front-end Structure

5.4.1 Live Data Feed

With the backend RESTful APIs the JavaScript queries the multiple databases. For the one-minute interval data stream, we query sMap for most recent minute and display data with a 5 second interval. Then the script continues fetching data from the database every 5 seconds. For the one-hour interval data stream, we display one sample per minute and add in a new data point each minute. Moving towards the day, week, month and year time intervals, we reference the lower-resolution databases we have created accordingly.

On the right side of the live data stream, we generate a breakdown of consumptions. Given not all devices are probed, we compute the consumption based on what we have available to present their percentage of the total. This is updated each time the pages area loaded. Since there is only one sensor available for water flow rate reading, we choose to present the data by different days of the week.

5.4.2 CSS styling

On top of all the great features and capabilities, the original goal of the project is to design a simple and user-friendly interface to display and convey information to the users. We have approached this problem by using many existing CSS styling. We have used Bootstrap, which was originally developed by Twitter. We also used SB-admin, which is an extension off of Bootstrap. All these CSS libraries and JavaScript libraries have enriched the user experiences with a more stylish look. All the graphs on the page are powered by Highcharts, a JavaScript interactive chart library. On top of all these existing libraries, we also wrote a few to extend the existing ones to fit our customized pages.

5.5 Information Displayed

Since both our sMap and reduced resolution database are very resourceful, we have many options on what to display. In a recent report [5], more consumers are shifting to become interested in receiving info from "low-touch" channels, such as email, applications and text messages.

5.5.1 Statistics

When it comes to the question about what to display on the dashboard, we took a step back and asked ourselves about what we would hope to see as consumers. We have identified the following key features: today's consumption thus far, yesterday's consumption and monthly totals at headline level. We decided to display a instantaneous power so that users can imagine how much an hour of the current level of consumption would end up on their bill. At the same time, we compute the daily total of the entire house measured at the breakers. To offer a comparison, we have put yesterday's total next to today's to encourage users to save energy. The page also offers a breakdown by percentage of each device's usage of the total amount.

5.5.2 Layout

The layout is hierarchical. Starting from the total of the month when loading the

homepage, then users can choose to view either power or water consumption. On both the power and water tab, users are provided with the scalable time series graph of consumption with time interval options of minute, hour, day week and year. The mapview is constructed on a separate tab so that the entire page area can be used for a floor plan with graphic overlay.

5.6 Household Consumption Map

To illustrate the energy usage in a more direct method, we decided to make a interactive map interface to better display the user consumption of resources. We have created a map of electricity usage and another map of water usage. We see this to be scalable for even building energy management. We have demonstrated how a single house can have an interactive map view to inform the owner where their most power hungry devices are. For commercial buildings, the map can be constructed in an exploded view style, where facilities can see and click on any level of the building.

5.6.1 Floor Plan Generation

A detailed floor plan was provided to us with a great amount of constructional information regarding the layout of the house. The floor plan is converted to a layout image with Google Sketchup. Google Sketchup allows designer to trace over existing images, such as scanned floor plans. In this project, a trace over is shown on the map view pages. For commercial buildings, Sketchup can also assist with the 3D drawing to have numbers of floors and even furniture.

5.6.2 Usage on Map

The real time usage is overlaid on the map of Cartesian coordinates 75x75 of grid size of 5px. Electronic device are represented by squares on the map. The boxes are shaded by a spectrum of color from green to red. Data series are formatted to have x and y location and usages values. Green is set to represent 0, corresponds to devices that are turned off. Red is set to the highest value of the data series. The color spectrum is linearly related to the values in the data series. Therefore, the more power hungry devices at any given moment would be red. This map is plotted every 5 seconds and continue to refresh as the page remains open.

A box pops up upon mouse over actions (computers) or clicks (computer and tablet). This box is referred as a “tool tip”. The tool tip provides a specific set of information for users and developers. The x and y location can be turned on or off for display during implementation and usage. The tool tip also displays the value of the consumption at the given moment. The tool tip can use HTML format, which allows more information asserted, such as usage alerts. The tool tip box is set to appear with a delay of one second for a better user experience.

5.7 Actuation

The actuation is handled by sending an http put request to one of several Philips hue lights upon the request of the user.

Sending notifications is also possible by sending the current estimated water/energy bill on some specified time interval.

6. CONCLUSION

Using the sMAP database alone was essentially impossible for obtaining the yearly/monthly/weekly usage information in an interactive web app, due to the large amount of high resolution data. We found a solution by creating a layered resolution database model, using relational database tables on top of the sMAP data. We were successful in doing this, however our solution is far from ideal, because it was difficult to implement, and it is difficult to use due to the disjointed nature using multiple databases. A better solution would be to have the layered resolution database model built into sMAP. This would ensure that the data layers are always in sync with each other, and the query system could be centralized, taking advantage of the low-resolution layers automatically. We believe this is desperately needed to make sMAP’s

query system more usable. As of right now, the query system is limited to searching 10,000 data-points, which limits the time-span of our queries to less than 3 hours for 1 Hz data. (It may be possible to increase this, but it wasn't clear how from the sMAP documentation, and there will be a larger delay with executing larger queries). A layered approach would allow sMAP to give results for arbitrarily large time-spans as long as appropriate low-resolution layers are included. This would reduce the amount of work that programmers must do, since they wouldn't need to implement this model themselves, and it would reduce the possibility of errors since accessing the data would be simplified.

6.1 Future Works

We believe the dashboard can be even better with a message box as illustrated on our page. We have envisioned a panel for utilities to send their message to their customers, such as alerts and tips. Researches have shown that the user behavior can change energy consumption of a household up to 30% of the total consumption. For example, utilities can ask consumers to turn up their thermostat or turn off other unused devices on a hot day to reduce their energy demand. Utilities can also show consumers how their consumptions are comparing to the area average, to further educate people on energy efficiency.

We also see the opportunity to make the panel into a feedback channel to utilities. Often, consumers don't have a choice of energy providers but it is still necessary to address their individual needs. Another drawback one may consider is that the map is specified to a floor plan. Since all the other components requires professional setup, we do not expect a general consumer to have the ability to recreate a floor plan digitally.

7. REFERENCES

[1] H.-x. Zhao and F. Magoules, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews,* vol. 16, no. 6, pp. 3586-3592, August 2012.

[2] Z. Yu, F. Haghighat, B. C. Fung, E. Morofsky and H. Yoshino, "A methodology for identifying and improving occupant behavior in residential buildings," *Energy,* vol. 36, no. 11, pp. 6596-6608, November 2011.

[3] M. LeMay, J. J. Haas and C. A. Gunter, "Collaborative Recommender Systems for Building Automation," in *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 2009.

[4] O. Aritoni and V. Negru, "A multi-agent recommendation system for energy efficiency improvement," *Communications in Computer and Information Science,* vol. 171, pp. 156-170, 2011.

[5] Accenture, "Actionable Insights for the New Energy Consumer," Accenture, 2012.