2011 IEEE/ACM Second International Conference on Cyber-Physical Systems

Efficient Sensing Matters a Lot for Large-scale Batteries

Hahnsang Kim and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2121, U.S.A.
Email: {hahnsang, kgshin}@eecs.umich.edu

***Abstract*—A large-scale battery pack that consists of hundreds or thousands of battery cells must be carefully monitored. Due to the divergence of cell characteristics, every cell should be monitored periodically and accurately. There are two important issues in monitoring large-scale packs. First, sensing the health condition of battery cells must be timely to capture the turning point at which the battery condition abruptly changes. Failure to capture such an important event can cause irreversible damage to the battery, especially when its State-of-Charge (SoC) is very low. Second, the more the hardware components are used, the higher the failure rate the system will suffer. The frequency of monitoring battery cells, thus, should be adjustable to the underlying load demand, considering the fact that a low load demand has a minute impact on the battery condition. We propose to address these issues via an adaptive monitoring architecture, called ADMON. ADMON lowers the sensing latency effectively, making it effective to enhance the tolerance of physical cell failures. ADMON consists of sensing, path-switching, and computing systems. The sensing system collects data from a battery-cell array. The path-switching system effectively connects a specific sensor and a micro-controller that is part of the computing system. The path-switching system is characterized by three exclusive types of topology: n-tree-based, cascaded, and parallel. The computing system is synergistically combined with the other two systems while three policies specified in the computing system are applied. The ADMON architecture is shown to outperform a non-adaptive monitoring system with respect to the battery life by 67%.**

***Keywords*-Battery management, battery monitoring, topology of multiplexers, moving average filter**

I. INTRODUCTION

Replacement of gasoline combustion vehicles with electric vehicles (EVs) makes dramatic impacts both environmentally and economically. It can significantly reduce greenhouse gas emissions that cause the global temperature to rise. For instance, the replacement of 77% of all transport miles with EVs can reduce carbon intensity by 94% over the 1990 numbers [39]. Meanwhile, we are faced with unprecedented challenges due to rising energy cost and its impact on the national competitiveness and security. To reduce the greenhouse gas emissions and the dependency on fossil fuels, it is imperative to harvest renewable energy that requires an efficient large-scale energy storage system.

Efficient battery management is key to large-scale energy storage systems. An effective and efficient battery management system (BMS) must extend the battery life as far as its constituent cells can last. However, the electrochemical interaction and reaction cause serially-connected battery cells to alter their characteristics, resulting in voltage-imbalance across the cells. Unbalanced voltages of the cells can place stress on the battery dynamics, accelerating cells' aging. As a result, the failure rate of the entire battery pack increases faster than that of individual cells. To prevent or slow down this process, cell-balancing—making the voltages of serially-connected cells equal or balanced—is an essential part of BMS capability. Also, accurate estimation of the battery SoC—that gauges the battery's energy level—is of great importance, leading to numerous efforts to reduce the SoC estimation error. For instance, Plett [26–28] proposed a method based on an extended Kalman filter for estimating SoC while adapting to dynamically-changing battery characteristics as cells age. Also proposed are various analytical methods based on fuzzy logic [32, 33] and neural networks [1, 3, 34]. All of these methods, including cell-balancing, require individual cells to be monitored accurately and cost-effectively.

The voltage of a lithium-ion battery for EVs linearly drops during the discharge period until it reaches a certain turning point in its SoC, and then falls steeply below a *cutoff voltage*. The cutoff voltage is battery chemistry-specific and used to prevent the battery from being *deeply-discharged*, which will otherwise cause an irreversible damage to the cell. *Over-charging* the battery can also incur such a damage. Over-charging and deep-discharging, therefore, must be avoided. One way to prevent this is to put additional thresholds on the charge–discharge range, as is done in the first-generation GM Volt that limits its available battery capacity to 50% by setting lower and upper bounds to 30% and 80% SoC, respectively [7].

As mentioned earlier, efficient battery monitoring is a fundamental issue that needs to be addressed theoretically and systematically. A large number of battery cells that form the battery (e.g., a 6,800-cell pack for Tesla S model [16] and a 300-cell pack for GM Volt [21]), to power embedded

978-0-7695-4361-1/11 $26.00 © 2011 IEEE
DOI 10.1109/ICCPS.2011.21

IEEE computer society

components such as an electric motor, need to be monitored periodically and/or on-demand. In other words, we must address the tradeoff between fault-tolerance in monitoring sensors and accuracy in assessing the battery condition, since a higher monitoring frequency imposes higher computation and communication workloads, and accelerates the process of physical wear-out on components.

We present an adaptive monitoring architecture, called ADMON. The ADMON architecture consists of sensing, path-switching, and computing systems. The sensing system collects data from sensors associated with battery cells. It can comprise as many as thousands of sensors matching battery cells. To cope with such a large number of sensors effectively, the path-switching system operates based upon an n-tree-based, cascaded, or parallel topology. One of these topologies is selected on the basis of requirements, e.g., the limited number of state vectors and multiplexers. The computing system is the smart software control of our adaptive monitoring. For adaptive monitoring, it executes three policies as needed. These policies improve the fault-tolerance of physical parts, e.g., sensors by singling them out as *ghost sensors* that are not temporarily used. The physical parts of ADMON are flexible enough to accommodate the capability of the smart control cyber systems while the software control effectively reflects circumstances exposed to ADMON. The ADMON architecture capitalizes on a synergistic combination of cyber and physical systems.

The main contributions of this paper are three-fold. First, we meet the need of a cyber-physical system approach for a large-scale battery system that manages thousands of battery cells. ADMON is applicable not only to such a battery system, but also to other large-scale sensing systems such as a renewable energy hybrid storage system. Second, ADMON gives a physical insight into a large-scale routing system by presenting the three types of routing topology, with ease of software control. These types can easily be combined as required. Third, the execution of the three policies in the software computing system is computationally inexpensive, making it effective to replace physical sensing. This software also improves the life of physical components.

The rest of the paper is organized as follows. Section II describes the design of ADMON that consists of the sensing, path-switching, and computing systems. Three topologies for the path-switching system and three policies for the computing system are presented. Section III presents the analysis of the overall latency in sensing and fault-tolerance. Section IV evaluates the performance of ADMON. We discuss the related work in Section V and conclude the paper in Section VI.

II. ADMON

This section first describes the architecture of the adaptive monitoring architecture, and then details its components.

A. The Architecture

An appropriate combination of hardware and software components is key to designing an architecture. A hardware system should be flexible enough to realize the capability bestowed by software, and a software system should be simple and effective enough for the hardware system to realize. With this taken into consideration, we build ADMON.

The ADMON architecture, as shown in Fig. 1, comprises three systems for: sensing, path-switching, and computing. The sensing system is responsible for collecting information on the battery-cell array in which battery cells are connected in series and/or in parallel. The array's information is collected via various sensors for voltage, temperature, and current. In particular, voltage (and current) can be measured at a cell- or module-level, depending on the specific design requirement. In the cell-level measurement, each cell-level unit (e.g., a cell) requires a dedicated voltage sensor, while in the module-level measurement, a voltage sensor per module (consisting of a group of cells) is required. In contrast, temperature sensors can be sparsely deployed across cells/modules. The distribution of sensors is important to the imbalance of temperatures that may cause battery behavior to sharply deteriorate. This distribution should be considered jointly with a cooling system that is beyond the scope of this paper. Similarly, current sensors are closely related to the arrangement of the inside and outside of the battery-cell array. A single current sensor would be sufficient for a chain of serially-connected cells, whereas parallel groups of cells/arrays may require multiple current sensors. In addition to three types of sensors, other sensors such as current-surge detection sensors can be deployed in the system.

Sensors can be implemented on a single chip such as the one in a chip manufacturer [18]. Although highly design-specific, it is worth looking at the design more closely. When all the sensors are used frequently and simultaneously, all-in-one chip can be an option. In this case, however, a fault-tolerance issue arises depending on a partial or total failure on such a sensor. When sensors are randomly chosen for use, they need to be implemented separately, which may increase the wiring and hence the cost.

The path-switching system is responsible for connecting a specific sensor and the corresponding micro-controller. As shown in Fig. 1, it is located in between the computing and sensing systems. Since the total number of sensors is likely greater than that of micro-controllers, the path-switching system resolves the disparity using 2^n-to-1 multiplexers (Muxes). With a state buffer given, the computing system determines the output of Muxes. Muxes are arranged in a tree structure by connecting the output of a Mux to an input of another. Three topologies for this will be detailed in the next section.

The computing system is the smart control of our adaptive

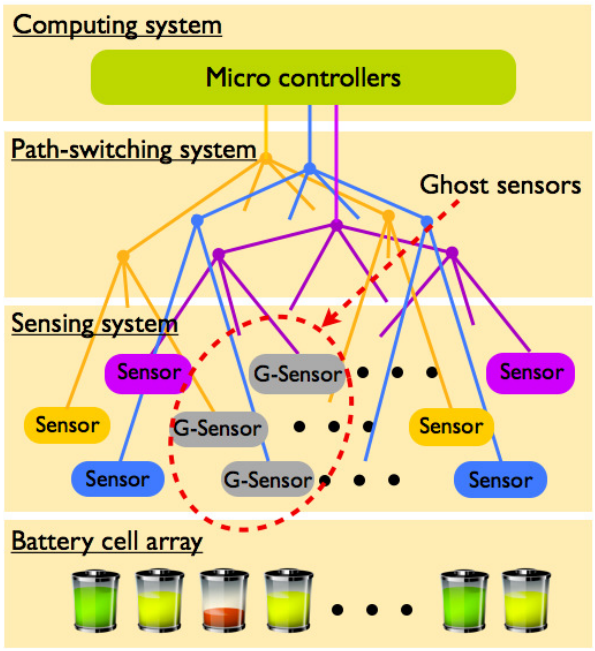


Figure 1. Layered-diagram of ADMON: The computing, path-switching, and sensing systems form the architecture. In particular, the sensing system comprises various types of sensor including "ghost" sensors. A battery-cell array is the physical arrangement of battery cells that are connected in series and/or in parallel. In principle, each cell needs a dedicated sensor.

monitoring. It consists of (distributed) micro-controller(s) (e.g., dsPIC24 [19]) and an adaptive scheduler thereon. A micro-controller is connected to a certain number of sensors via Muxes in the path-switching system, and determines the number of Muxes based on a specific topology, configuring their state vector. On the other hand, the adaptive scheduler determines which sensor to be read at what frequency because the sensing frequency depends on the type of sensor used. It also varies with the underlying load demand. For instance, when voltage is to be measured from voltage sensors with a low load demand, we may skip reading some of these sensors, which we call *ghost* sensors. As long as the load demand is below a certain threshold, the ghost sensors serve as backups in case certain sensors fail. Ghost sensors can be considered redundant from a manufacturer's perspective. Thus, the number of necessary ghost sensors gives a practical insight into the design of the monitoring architecture, which is the main object of this paper.

The ADMON architecture is based on a top-down view. It has three advantages over a bottom-up approach. First, the top-down approach makes adaptive monitoring most effective. Unlike other variables like temperature, the voltage of a certain cell is to be monitored together with other cells'. Otherwise, adaptive monitoring becomes as ineffective as bruce-force monitoring. Second, this architecture is cost-effective. A bottom-up approach requires a computation-intensive module for each battery cell. By contrast, the top-down adaptive monitoring reduces the number of hardware components necessary for sensing, referred to as ghost sensors. Third, the architecture is easier to make each battery cell self-checking/aware. A bottom-up approach may also make each cell less reliable by adding more circuits to the cell.

In what follows, we briefly investigate key hardware components for each constituent system from a practical perspective, and then compare the topology along with an operating algorithm.

B. The Sensing and Computing Systems

We consider three of sensors: voltage sensors $S_{vs} = \{vs_1, vs_2, \cdots, vs_m\}$; current sensors $S_{cs} = \{cs_1, cs_2, \cdots, cs_n\}$; and temperature sensors $S_{ts} = \{ts_1, ts_2, \cdots, ts_o\}$. Assuming that the characteristics of same-type sensors are identical, the latency of each type in configuring Muxes and retrieving values is denoted by t_{vs}, t_{cs}, and t_{ts}.

The authors of a survey [29] have made a useful comparison of various micro-controllers. In particular, the number of digital I/O lines is a dominant concern in a large-scale monitoring system. Although micro-controllers vary with manufacturers, their I/O lines are reported to range from 4 to 100. For instance, 8-bit PIC micro-controllers have 4 or 25 lines available, and 32-bit micro-controllers of the same family come with 85 I/O lines [20]. The latter model, however, is 12 times more expensive than the former. Yet, not only the cost but also the frequency/MIPS of micro-controllers needs to be considered while the required computational power of micro-controllers relates to the path-switching system that will be described later.

Ghost sensors are managed with three policies applied, based on which a set of ghost sensors is created, updated, or removed. First, sensors of the same type are colored according to the battery arrangement. Sensors for a group of serially-connected cells are assigned same color, while the cells of a different group are labeled with a different color. The reason for this coloring is that cells within a group interact with, and affect, each other somewhat in a similar fashion, as they age. Although this coloring policy is requirement-specific, it is effective for classifying sensors. A simple implementation of this policy is described in pseudo-code as follows.

Input: ns: # of cells in series;
np: # of groups in parallel;
Output: V: $ns \times np$ sensor matrix;
for $m = 1 : np$
  for $l = 1 : ns$
    $V(m,l) \leftarrow m$;

Second, the frequency at which to read sensors is proportional to the corresponding cells' discharge (and charge) rate.

This policy reflects the fact that a higher discharge/charge rate implies a faster chemical reaction inside the cell. This policy, however, is applied in a *steady zone* in which the SoC of a battery cell ranges from a lower bound (e.g., 10∼20%) to an upper bound (e.g., 80∼90%). The change in battery characteristics outside of the zone is nonlinear, requiring cells to be monitored at a high rate. The concept of using the zone is effective in voltage sensing.

The discharge rate is subject to the time-varying load demand, which is somewhat predictable. So, when the change in the discharge rate follows a predicted pattern associated with a sensor, the sensor is considered as a ghost sensor. To predict the pattern of a time-varying load demand, we apply a weighted moving average filter [22]:

$$y(k) = \sum_{i=0}^{k-1} a_i y(i), \quad \text{where } \sum_{i=0}^{k-1} a_i = 1. \tag{1}$$

The weighted moving average filter is simple and inexpensive to implement, and good for the prediction purpose. Unless an estimate deviates from a measured value by more than a certain threshold (i.e., $|y(k) - b| > \delta$, where b and δ are a measured value and the threshold), the estimated value is used without reading the sensor. In such a case, it is imperative that computing Eq. (1) be less expensive than reading a sensor.

There are various complex moving average filers [5] and applications using them [11, 12]. Moreover, using the Kalman filter [38] is an option. The Kalman filter has also been applied for not only tracking mobile agents [10, 41] but also estimating the SoC of a battery [15, 26–28, 36, 37].

Third, the similarity between different groups of cells is used to predict the change in battery conditions. For instance, when two groups are connected in parallel, both groups are likely to behave the same within a tolerance range. So are cells within each group. Hence, monitoring one group/cell would suffice to capture the behavior. The similarity between two groups is measured by applying χ^2-distance [4]:

$$\chi^2(V,\ V') \ = \ \sum \frac{(V \ - \ V')^2}{(V \ + \ V')}, \tag{2}$$

where V and V' are buffers for sensed data from the first and second groups of cells, respectively. When all the data in V match those in V', the value of χ^2 becomes zero. That is, the higher value of χ^2, the greater dissimilarity between the two groups.

The χ^2-distance has been used in diverse areas, such as scene-change detection in image sequences [6, 23] and anomaly detection [12, 40]. In addition, experimental results [13] show that the use of the χ^2-distance reduces the amount of computation over a widely-used technique, i.e., the Bhattacharyya distance [31].

C. The Path-switching System

A Mux in the path-switching system consists of a 2^k-to-1 encoder with a k-bit state vector. That is, a k-state vector and 1-bit output allow a micro-controller to be connected to 2^k sensors. When the number m of sensors is greater than 2^k, either combining multiple Muxes or deploying additional micro-controllers will be an option. In this combination, $\lceil \frac{m}{2^k} \rceil$ Muxes are required, and the number of state vectors also increases accordingly. This increase requires more input lines from a micro-controller. This eventually requires more micro-controllers. In this paper, however, we do not consider the issue of designing a new Mux that may accommodate more sensors with less logic gates therein.

We now present three path-switching topologies: n-tree-based, cascaded, and parallel.

1) The n-tree-based topology: Each input of a Mux is connected to the output of another where $n = 2^k$. This topology requires an $n \cdot k$-bit state vector to manipulate Muxes unless optimized. We propose to share a k-bit state vector with n Muxes, resulting in a $2k$-bit state vector in which n Muxes behave identically. For instance, suppose a 4-bit Mux that includes a 2-bit state vector is used, as shown in Fig. 2. Then, a path is routed by calculating:

$$\mathbf{O} = I_0 \bar{s_2} \bar{s_3} + I_1 \bar{s_2} s_3 + I_2 s_2 \bar{s_3} + I_3 s_2 s_3, \tag{3}$$

where I_i is the output of each Mux on the bottom, which is recursively calculated, and s_i is the $(i+1)$-th bit on the state vector.

The state vector is set from top to bottom. First, given the j-th sensor to be read, the input of the Mux on the top is determined as:

$$p = \lceil \frac{j}{n} \rceil. \tag{4}$$

When $n = 4$, S_{23} (which represents a value for the 3rd and 4-th bits on the state vector) is set to p. Then, the other bits for the p-th Mux is configured by

$$S = j - n(S_{23} - 1). \tag{5}$$

That is, $S_{01} = S$. Fig. 3 shows an algorithm for the above calculations. This algorithm can cope with large-scale sensors, and scale to any n-bit Muxes of this type.

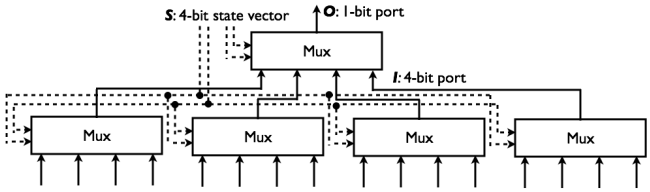


Figure 2. The n-tree-based topology of the path-switching system, where $n = 4$.

2) The cascaded topology: The last input bit of a Mux is connected to the output of another, resulting in a chain of Muxes as shown in Fig. 4. Individual Muxes are configured with their own state vectors, which is straightforward. Given the j-th sensor to be read, a computation algorithm for the

Input: n: # of output bits;
h: # of layers;
j: a designated sensor's location;
Output: S: layers of $2k$-bit state vectors;
for $m = 1 : h-1$
  $p \leftarrow$ Eq. (4);
  if $p \geq n$
    $S \leftarrow [S,$ Eq. (5)$]$;
  else
    $S \leftarrow [S,\ p]$;
  $j = p$;
$S \leftarrow [S,\ j]$;

Figure 3. Computation algorithm for the n-tree-based topology

cascaded topology pinpoints the Mux that is connected to the sensor by calculating

$$p = \lceil \frac{j}{n-1} \rceil. \tag{6}$$

The p-th Mux is then configured with the state vector set by

$$S = j - (n-1)(p-1), \tag{7}$$

while the state vector of each Mux from the first to the $(p-1)$-th is set to n. Then, state vectors from the $(p+1)$-th

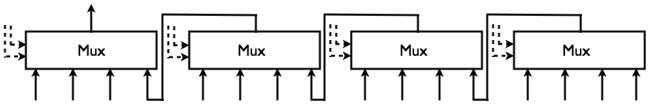


Figure 4. Cascaded topology of the path-switching system

to the last Mux are considered as *don't care*. Fig. 5 shows the computation algorithm described above.

Input: n: # of output bits;
j: a designated sensor's location;
Output: S: an $n \cdot k$-bit state vector;
$p \leftarrow$ Eq. (6);
for $m = 1 : p-1$
  $S \leftarrow [S,\ n]$;
$S \leftarrow [S,$ Eq. (7)$]$;

Figure 5. Computing algorithm for the cascaded topology

3) The parallel topology: Similar to the n-tree-based topology, the parallel topology, shown in Fig. 6, shares a state vector with n Muxes. However, it allows one to simultaneously read multiple sensors. This topology is effectively used in conjunction with specific battery arrangement. For instance, each Mux is associated with a group of serially-connected cells, and then these groups are connected in parallel. In such a case, cells at the same position in each group can be monitored in a timely manner. Also, the parallel topology allows the overhead for frequent switchings to diminish with a factor of n, especially making a great impact on the monitoring at a high discharge rate.

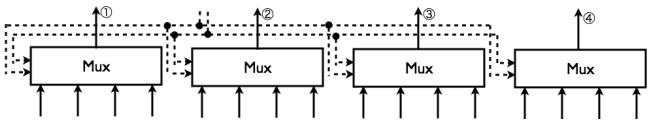


Figure 6. Parallel topology of the path-switching system

In contrast, an input of each Mux can be associated with a cell within a group of serially-connected cells. Then, cells in the group can be monitored simultaneously. This association is very effective for *cell-balancing* in which voltages across the cells are kept identical within a tolerance range during their charging (and discharging).

Fig. 7 shows a computation algorithm for the parallel topology. This algorithm allows one to read up-to n numbers of sensors, ultimately pinpointing the specific Mux associated with a certain sensor.

Input: n: # of output bits;
j: a designated sensor's location;
Output: S: a k-bit state vector;
p: a designated Mux's location;
$p \leftarrow$ Eq. (4);
$S \leftarrow$ Eq. (5);

Figure 7. Computation algorithm for the parallel topology

III. Performance Analysis

Let us consider an $m \times n$ battery pack that consists of m arrays connected in parallel, each of which is n serially-connected cells. At the time of the deployment of the pack, all battery cells therein are assumed identical, but their characteristics drift apart from one another due to interactions between neighboring cells. Since we want to balance the cell characteristics both within and outside each array, every cell within an array needs to satisfy the following condition: $|y(k) - b| \leq \delta$ (i.e., Eq. (1)) with 1-p_s, where p_s is the probability that a cell's voltage equals a specified value (e.g., a mean/median value) within a certain range. Assuming that the characteristics of cells develop independently, the event of cell-balancing is a Bernoulli trial, resulting in the expected number of cells that converge to the balanced state:

$$E[I_c] = n \cdot p_s. \tag{8}$$

Likewise, the event of *array-balancing* is also a Bernoulli trial, resulting in the expected number of arrays that converge to the balanced state:

$$E[I_a] = m \cdot p_s^n. \quad (9)$$

The similarity policy presented in Section II-B is assumed to be applied only when all cells within a group are balanced.

When $p_s > 0$ adaptive monitoring is in operation, while brute-force monitoring when $p_s = 0$. As mentioned before, Eq. (1) is computationally less expensive than reading a sensor, i.e., configuring a Mux and buffering the corresponding value.

Suppose $p\{|X - b| \leq \delta_c\}$ during a monitoring interval where X is exponentially distributed with parameter λ_c. Then, the probability of successful prediction at the cell-level over time T is also exponentially distributed with parameter λ_c. From this, we can derive the duty cycle of a sensor, i.e., $p_s \cdot \lambda_c / T$. If events of applying the similarity policy at the sensor-level are exponentially distributed with parameter λ_a, the duty cycle can be reduced further with the policy applied and becomes $p_s^n \cdot \lambda_a / T$. This way, we can deduce the duty cycle of each sensor, and this can be generalized for an arbitrary number of sensors of various types.

IV. Evaluation

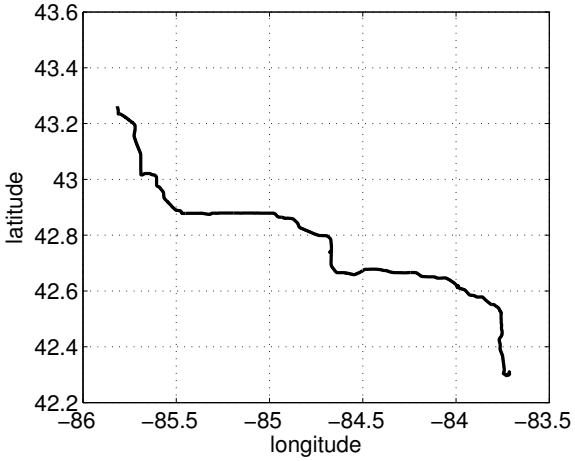
Our goal is to enhance the monitoring of a large-scale battery pack by effectively reducing the sensing frequency, ultimately improving the fault-tolerance of sensors associated with individual battery cells. To evaluate the efficacy and efficiency of the ADMON architecture, the metrics we use include the duty cycle and the reduction in discrepancies between measured and estimated values.

We first describe the evaluation setup and then, based on the metrics, demonstrate the ADMON's superior performance over a bruce-force monitoring scheme.

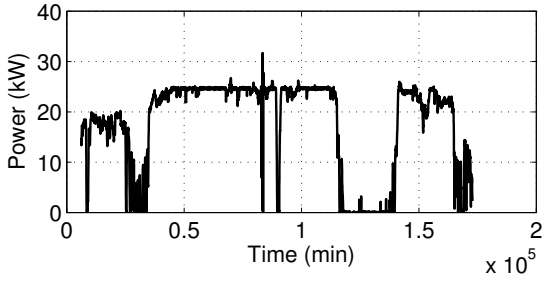
A. Evaluation Setup

For simplicity, we configure a battery pack as 64S4P (64 serially-connected cells and 4 parallel-connected groups/modules). Also, one sensor of a kind is required for each cell, resulting in a total of 625 sensors per kind. We use 4-to-1 Muxes to build the path-switching system. So, the *n*-tree, cascaded, and parallel topologies require per module 21, 21, and 16 Muxes, respectively. Pairs of the size of a bitwise state vector and the number of an output port for each topology are (6, 1), (42, 1), and (2, 16).

Load-demand profile: We measured locations in every 2 seconds from a GPS receiver while driving a vehicle. Its route, as shown in Fig. 8-(a), constitutes 164 miles (264 kilometers). From these data, we extract a pattern of load demands as well as the speed. The pattern is represented as change in the vehicle's power demand according to the route, as shown in Fig. 8-(b). The vehicle's power demand is calculated in proportion to its speed with maximum 111 kW (equivalent to 150 horse power). The corresponding pattern is estimated by applying a moving average filter (with k set to 10).



(a) Real on-the-road traces

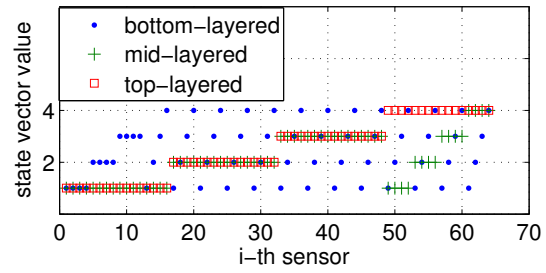


(b) Power demand

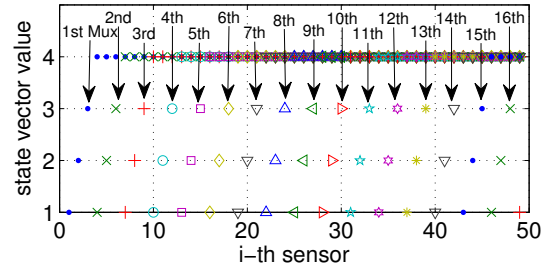
Figure 8. Power load demands based on real on-the-road traces

B. Evaluation Results

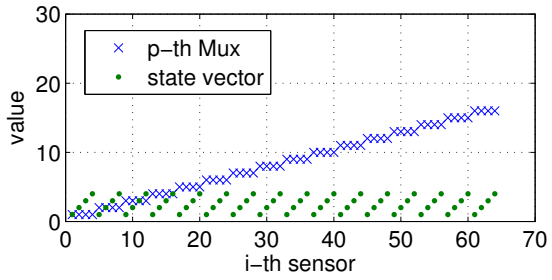
1) The path-switching system operates effectively: The path-switching system in ADMON is based exclusively on the *n*-tree, cascaded, or parallel topologies. The *n*-tree-based topology shares a state vector with Muxes in the same layer. So, the size of the state vector used for all Muxes is proportional to the height of the tree. For instance, as shown in Fig. 9-(a), the three-layered tree topology requires a 6-bit state vector. The cascaded topology, on the other hand, requires as many state vectors as the number of Muxes used. Also, an input of each Mux is used to connect itself to another Mux. Thus, when a certain Mux is used, the rest of Muxes in the chain are not considered being in use. For instance, in Fig. 9-(b), in reading the 4-th sensor, only the 1st and 2nd Muxes are actively used. Similar to the *n*-tree-based topology, the parallel topology shares a state vector with Muxes and yet is one-layer-based. This requires as many outputs as Muxes used but allows one to simultaneously read sensors.



(a) n-tree-based topology



(b) cascaded topology

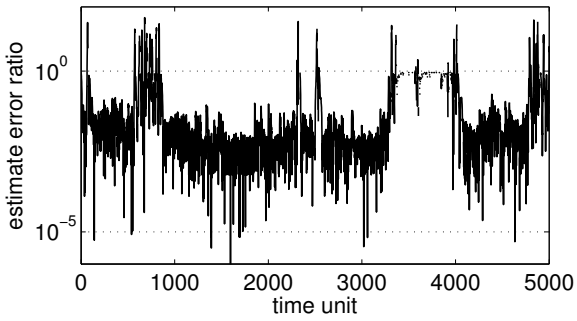


(c) parallel topology

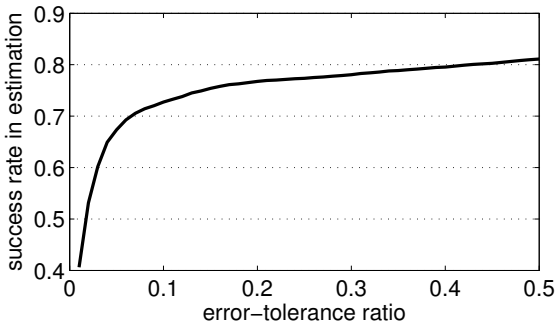
Figure 9. Configuration of state vectors for the three types of topology

2) The moving average filter operates effectively: The moving average filter smooths out local jitters, yielding a pattern of changes. This pattern reflects future load demands. An estimate is made at every interval, i.e., 2 seconds, and discrepancies between estimation and observation are to be reduced. Fig. 10-(a) plots the ratio of estimates to the real traces shown in Fig. 8-(b). Overall, the estimation error is below 1%, while zero values from the observation incur relatively more noise. More precisely, the ratio of success in estimation is contingent upon the error-tolerance ratio, δ. For instance, when $\delta = 0.05$, the moving average filter achieves a 67% success ratio, while achieving a 80% success ratio for $\delta = 0.5$. The success ratio follows the shape of an exponential distribution, as shown in Fig. 10-(b), indicating a very narrow spectrum of error-tolerance ratio.

3) The battery arrangement affects the performance gain: The higher probability of success in estimation, the better fault-tolerance performance of sensors. We calculate the performance gain per cell using Eqs. (8) and (9). Intuitively, the success ratio dominates the performance gain, and yet, applying the similarity policy using Eq. (2) also improves it.



(a) Estimation error



(b) Success ratio in estimation

Figure 10. Moving average filter-based estimation

For instance, as shown in Fig. 11, given a 67% success ratio, for a small number of serially-connected cells (e.g., $n = 8$ to 10), the performance is enhanced by 6%. This, however, becomes quickly ineffective as the value of n increases. On the other hand, increasing the number of modules (i.e., m) also improves the performance gain. That is, as the value of m is increased by 2, the performance gets better by a factor of 1.015.

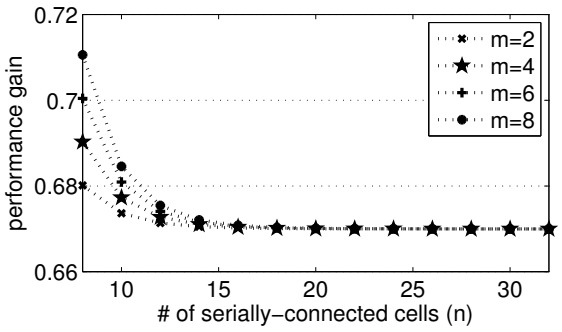


Figure 11. Performance gain per cell with respect to the battery arrangement

V. Related Work

Large-scale monitoring systems are often found in data centers for the purpose of debugging, performance measurement, and operational monitoring. For instance, Chukwa

[2] is a system that monitors 2,000 nodes, collecting 5 to 6MB of data per second from each node, and making collected data available for processing within 10 minutes. This monitoring, however, generates high traffic volume. To reduce such heavy traffic, hierarchical caching of data on proxy servers could be an option. ADMON is similar to Chukwa with respect to its architecture and operational management. There are also other monitoring systems, such as Splunk [9], Astrolabe [30], Pier [8], Ganglia [17], and Google's System Health Infrastructure [25].

Although the path-switching system of ADMON is application-specific, it can be generalized to build reconfigurable arrays. Concepts for reconfigurable arrays are based on various applications. For instance, multiplexer-based multipliers [24] are relevant to ADMON. An array of smaller multipliers rather than a monolithic block with a large bit width is advantageous. A flexible approach with reconfigurability at runtime can help save hardware resources and improve the efficiency of arithmetic operations. Such multiplexer-based circuits can be implemented in the form of a gate-array architecture [35], which contains a base row having 4 alternating P- and N-channel transistor rows. This architecture is efficient, particularly when it is used to create serial multiplexer-based circuits. There is also a variant of multiplexer-based architecture that provides high-density and low-power dissipation [14].

VI. CONCLUSION

An efficient monitoring mechanism (i.e., a cyber system) can allow hardware (battery cells and vehicle) components (i.e., a physical system) to work effectively, thereby improving their fault-tolerance. We presented the ADMON architecture to adaptively monitor thousands of battery cells within a large-scale battery system. Its physical parts are flexible enough to exploit the capability of smart software controls, and the software controls effectively communicate with, and adapt to, physical circumstances. Battery arrangement is found important to enhance the monitoring, resulting in three monitoring policies that are part of the computing system. With these policies, we have applied a moving average filter to estimate load demands, which is computationally inexpensive. Besides, the path-switching system operates on the basis of *n*-tree-based, cascaded, or parallel topology along with software controls. This system effectively copes with a large number of sensors, facilitating their combination. The cyber-physical systems approach presented by ADMON allows ADMON to lend itself to various applications that manage a large number of sensors.

REFERENCES

[1] A. Affanni, A. Bellini, C. Concari, G. Franceschini, E. Lorenzani, and C. Tassoni. Ev battery state of charge: neural network based estimation. In *International Electric Machines and Drives Conference (IEMDC)*, volume 2, pages 684 – 688. IEEE, July 2003.

[2] Jerome Boulon, Andy Konwinski, Runping Qi, Ariel Rabkin, Eric Yang, and Mac Yang. Chukwa: A large-scale monitoring system. In *Cloud Computing and Its Applications*, 450 North Cityfront Plaza Drive, Chicago, IL 60611, 2008.

[3] C.C. Chan, E.W.C. Lo, and Shen Weixiang. The available capacity computation model based on artificial neural network for lead-acid batteries in electric vehicles. *J. of Power Sources*, 87(1-2):201–204, 2000.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. ISBN 0-471-05669-3. Wiley-Interscience, second edition, 2001.

[5] Robert D. Edwards and John Magee. *Technical analysis of stock trends*. ISBN 0814406807. AMACOM, New York, NY, 8th edition, 2001.

[6] R.M. Ford, C. Robson, D. Temple, and M. Gerlach. Metrics for scene change detection in digital video sequences. In *ICMCS*, pages 610–611, Los Alamitos, CA, USA, 1997. IEEE.

[7] GM. Chevy volt battery pack operation. http://gm-volt.com/2007/08/29/latest-chevy-volt-battery-pack-and-generator-details-and-clarifications/.

[8] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, Loo Scott Shenker, and Ion Stoica. Querying the internet with pier. In *VLDB*, pages 321–332, Berlin, Germany, 2003.

[9] Splunk Inc. It search for log management, operations, security and compliance. http://www.splunk.com/base/Documentation.

[10] Hahnsang Kim and Kang G. Shin. On dynamic reconfiguration of a large-scale battery system. In *RTAS*, pages 87–96, San Francisco, CA, U.S., Apr. 2009. IEEE.

[11] Hahnsang Kim, Kang G. Shin, and Padmanabhan Pillai. MODELZ: Monitoring, detecting, and analyzing energy-greedy anomalies and mobile malware. *TMC in press*, 2010.

[12] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting energy-greedy anomalies and mobile malware variants. In *MobiSys*, pages 239–252, Breckenridge, Colorado, USA, Jun. 2008. ACM.

[13] Branislav Kisačanin, Vladimir Pavlovic, and Thomas S. Huang, editors. *Real-time vision for human-computer interaction*. ISBN 387276971. Springer, 1 edition, 2005.

[14] Robert Landers, Shivaling Mahant-Shetti, and Carl Lemonds. A multiplexer-based architecture for high-density, low-power gate arrays. *IEEE Jrl. of Solid-State Circuits*, 30(4):392–396, Apr. 1995.

[15] Jaemoon Lee, Oanyong Nam, and B.H Cho. Li-ion battery soc estimation method based on the reduced order extended kalman filtering. *J. of Power Sources*, 174(1):9–15, 2007.

[16] John Markoff. Nyt: Pursuing a battery so electric vehicles can go the extra miles. http://www.nytimes.com/2009/09/15/science/15batt.html?scp=1\&sq=electricvehicletesla20ibm\&st=cse, Sep. 2009.

[17] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia the gangelia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[18] MAXIM. MAX11068. http://www.maxim-ic.com/quick_view2.cfm/qv_pk/5523/t/al, 2010.

[19] Microchip. PIC24FJ128GA010 data sheet. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en024805.

[20] Microchip. Pic32MX320F128l. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532439, 2010.

[21] General Motors. Latest chevy volt battery pack and generator details and clarifications. http://gm-volt.com/2007/08/29/latest-chevy-volt-battery-pack-and-generator-details-and-clarifications/, Aug. 2007.

[22] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[23] N.V. Patel and I.K. Sethi. Compressed video processing for cut detection. *Vision, Image and Signal Processing*, 143(5):315–323, October 1996.

[24] Oliver A. Pfander, Roland Hacker, and Hans-Jorg Pfleiderer. A multiplexer-based concept for reconfigurable multiplier arrays. In *FPL*, LNCS 3203, pages 938–942. Springer-Verlag Berlin Heidelberg, 2004.

[25] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. In *Conference on File and Storage Technologies (FAST)*, pages 17–29, San Jose, CA, 2007. USENIX.

[26] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs part 1. background. *J. of Power Sources*, 134(2):252–261, 2004.

[27] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs part 2. modeling and identification. *J. of Power Sources*, 134(2):262–276, 2004.

[28] Gregory L. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs part 3. state and parameter estimation. *J. of Power Sources*, 134(2):277–292, 2004.

[29] Kai Qian, David den Haring, and Li Cao. *Survey of Popular Microcontrollers*. Springer, July 2009.

[30] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM TOCS*, 21(2):164–206, 2003.

[31] C. Reyes-Aldasoro and A. Bhalerao. The bhattacharyya space for feature selection and its application to texture segmentation. *Pattern Recognition*, 39(5):812–826, May 2006.

[32] Alvin J. Salkind, Craig Fennie, Pritpal Singh, Terrill Atwater, and David E. Reisner. Determination of state-of-charge and state-of-health of batteries by fuzzy logic methodology. *Journal of Power Sources*, 80(1-2):293–300, July 1999.

[33] Pritpal Singh, Craig Fennie Jr., and David Reisner. Fuzzy logic modelling of state-of-charge and available capacity of nickel/metal hydride batteries. *J. of Power Sources*, 136(2):322–333, 2004.

[34] Michael Scott Sullivan, Ronald David Brost, Yaobin Chen, and Russell Carley Eberhart. Method and apparatus for determining battery state-of-charge using neural network architecture. http://www.freepatentsonline.com/6064180.html, May 2000.

[35] Dzung J. Tran and Mark W. Acuff. Gate array architecture for multiplexer based circuits. Technical Report 808249, United State Patent and Trademark Office, 1998.

[36] M. Urbain, S. Rael, B. Davat, and P. Desprez. State estimation of a lithium-ion battery through kalman filter. In *Power Electronics Specialists Conf.*, pages 17–21. IEEE, June 2007.

[37] A. Vasebi, S.M.T. Bathaee, and M. Partovibakhsh. Predicting state of charge of lead-acid batteries for hybrid electric vehicles by extended kalman filter. *J. of Power Sources*, 49(1):75–82, 2008.

[38] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, UNC at Chapel Hill, July 2006.

[39] Christopher Yang, David McCollum, Ryan McCarthy, and Wayne Leighty. Meeting an 80% reduction in greenhouse gas emissions from transportation by 2050: A case study in california. *Transportation Research Part D: Transport and Environment*, 14(3):147–156, 2009.

[40] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Eng. Int'l*, 17(2):105–112, Oct. 2001.

[41] Zainab R. Zaidi and Brian L. Mark. Real-time mobility tracking algorithms for cellular networks based on kalman filtering. *TMC*, 4(2):195–208, Mar. 2005.