Self-Constructive High-Rate System Energy Modeling for Battery-Powered Mobile Systems

Mian Dong and Lin Zhong

Rice University
System Energy Model

\[ y(t) = f(x_1(t), x_2(t), \ldots, x_p(t)) \]

Response \( y(t) \): Energy consumed by the system in \( t \)

Predictors \( x_i(t) \): System status variables in \( t \)
Rate \( \frac{1}{t} \)

- 0.01Hz
- 1Hz
- 100Hz
A High-Rate Energy Model is needed to provide an energy reading at each OS scheduling interval 10ms.
Model Construction

\[ t = t_1 = t_2 = \ldots = t_n \]

\[ X(t) \]

\[ Y(t) \]

Regression
Model Construction

Linear Model:

\[ y(t) = \beta_0 + \beta_1 x_1(t) + \ldots + \beta_p x_p(t) \]

\[ \hat{\beta} = \text{argmin}_\beta ( \| Y(t) - [1 \ X(t)]\beta \|_2 ) \]
Model Construction

t=t_1=t_2=...=t_n

Target System

Data Acquisition Equipment

Host PC

Linear Model:

\hat{y}(t) = \hat{\beta}_0 + \hat{\beta}_1 x_1(t) + ... + \hat{\beta}_p x_p(t)

err(t_i) = \frac{\hat{y}(t_i) - y(t_i)}{y(t_i)} \quad \text{Mean Absolute Root-Mean-Square}
What are the limitations?
External Devices for energy measurement
Deep Knowledge for predictor collection

www.nokia.com
Exclusive Model for a specific platform
Fixed Model for all instances of the same platform
Dependencies of system energy models on Hardware & Usage suggest “personalized” models be constructed for a mobile system.
Self-Constructive System Energy Modeling
Battery Interface
State-of-the-art battery Interfaces are Low-rate/Inaccurate

<table>
<thead>
<tr>
<th>Max Rate</th>
<th>N85</th>
<th>T61</th>
<th>N900</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Rate</td>
<td>4Hz</td>
<td>0.5Hz</td>
<td>0.1Hz</td>
</tr>
<tr>
<td>Accuracy</td>
<td>67%</td>
<td>82%</td>
<td>58%</td>
</tr>
</tbody>
</table>

Accuracy = $100\% - \text{Root\_Mean\_Square(Instant\_Relative\_Error)}$
Errors in battery interface readings are **Non-Gaussian**
Low-Rate/Inaccurate Battery Interface

Statistical Learning

High-Rate/Accurate System Energy Model
Averaged battery interface readings have Higher Accuracy but Even Lower Rate.
Linear models are independent on Time.
1. Model Molding

\[ Y(t_{VL}) \rightarrow \cdots \rightarrow Y(t_L) \rightarrow \hat{Y}(t_H) \]

\[ X(t_{VL}) \rightarrow \cdots \rightarrow X(t_H) \rightarrow \hat{\beta} \]

0.01Hz \quad 1Hz \quad 100Hz
RMS of Relative Error

Rate (Hz)

Battery Interface
Model Molding improves rate

![Graph showing RMS of Relative Error vs. Rate (Hz)]

- **Battery Interface**
- **Molded Model**
2. Predictor Transformation

\[ x_1(t), x_2(t), \ldots, x_p(t) \]

\[ z_1(t), z_2(t), \ldots, z_L(t) \quad L \leq p \]
PCA improves accuracy

![Graph showing RMS of Relative Error vs Rate (Hz)]

- Battery Interface
- Molded Model
- Molded Model + PCA
3. Total-Least-Square

\[ y = f(x) \]

- Training data points

- \( y_j \)
- \( r_j \)
- \( d_j \)
- \( x_j \)
- \( \Delta x_j \)
TLS improves accuracy at high rate
Sesame

Model Constructor

Predictor transformation → Model molding

Data Collector

Predictors → Responses

Stats readings → Energy readings

Application specific predictors

STATS
Operating System
Bat I/F
Implementation

N900

T61
Sesame is able to generate energy models with a rate up to 100Hz

<table>
<thead>
<tr>
<th></th>
<th>T61</th>
<th>N900</th>
</tr>
</thead>
<tbody>
<tr>
<td>1Hz</td>
<td>95%</td>
<td>86%</td>
</tr>
<tr>
<td>100Hz</td>
<td>88%</td>
<td>82%</td>
</tr>
</tbody>
</table>

Accuracy = 100% – Root_Mean_Square(Instant_Relative_Error)
Field Study

Day 1-5: Model Construction
Day 6: Model Evaluation
Models were generated within 15 hours

![Graph showing error over time for different users and frequencies.](image)
Sesame is close to State-of-the-Art without any external assistance

![Graph showing average absolute relative error vs. rate (Hz) for different systems: Sesame, PowerTutor, PowerBooster, and FSM.]

- **Sesame**: Close to state-of-the-art without external assistance.
- **PowerTutor**: CODES+ISSS’10
- **PowerBooster**: CODES+ISSS’10
- **FSM**: Eurosys’11
Sesame is able to construct models of high accuracy because of

1. Sophisticated Statistical Methods
2. Capability to Adapt Models
Sesame is a high-rate/accurate Virtual Power Meter
and creates new opportunities in Energy Optimization & Management
Software Optimization

\[ y(t) = \beta_0 + \beta_1 x_1(t) + \ldots + \beta_p x_p(t) \]

“Knob” provided by target software
Energy Accounting

\[ y(t) = \beta_0 + \beta_1 x_1(t) + \ldots + \beta_p x_p(t) \]

\( n \) Processes
Energy Accounting

\[ y(t) = \beta_0 + \beta_1 x_1(t) + \ldots + \beta_p x_p(t) \]

\[
\begin{align*}
  x_1(t) &= x_{1,1}(t) + \ldots + x_{1,n}(t) \\
  &\vdots \\
  x_p(t) &= x_{p,1}(t) + \ldots + x_{p,n}(t)
\end{align*}
\]
Energy Contribution by Process $j$

$$y_j(t) = \beta_1 x_{1,j}(t) + \ldots + \beta_p x_{p,j}(t)$$
Sesame can be also used for **Servers and Workstations**
Conclusions

• Self-Modeling is necessary to adapt to the changes in hardware and usage

• Statistical methods help to construct high-rate / accurate models from low-rate/ inaccurate battery interfaces

• Sesame creates new opportunities in system energy optimization and management
Sesame is close to optimal

![Graph showing the RMS of Relative Error vs. Rate (Hz) for Bat I/F, Sesame, and External. The graph indicates that Sesame is close to optimal performance.]
Why user experience is good?

Latency (ms)

- **N900 (pca+syscall)**
  - Response collection: 1
  - Predictor collection: 1
  - Model calculation: 1
- **N900 (pca)**
  - Response collection: 1
  - Predictor collection: 2
- **N900 (original)**
  - Response collection: 1
  - Predictor collection: 3
  - Model calculation: 1
- **T61 (pca+syscall)**
  - Response collection: 1
  - Predictor collection: 1
- **T61 (pca)**
  - Response collection: 1
- **T61 (original)**
  - Response collection: 1
Models built with different apps estimate differently by \(30\%\).
Why High rate is hard?
What is the limitation of linearity

- USENIX paper
Why Sesame is better for laptop than smartphone?