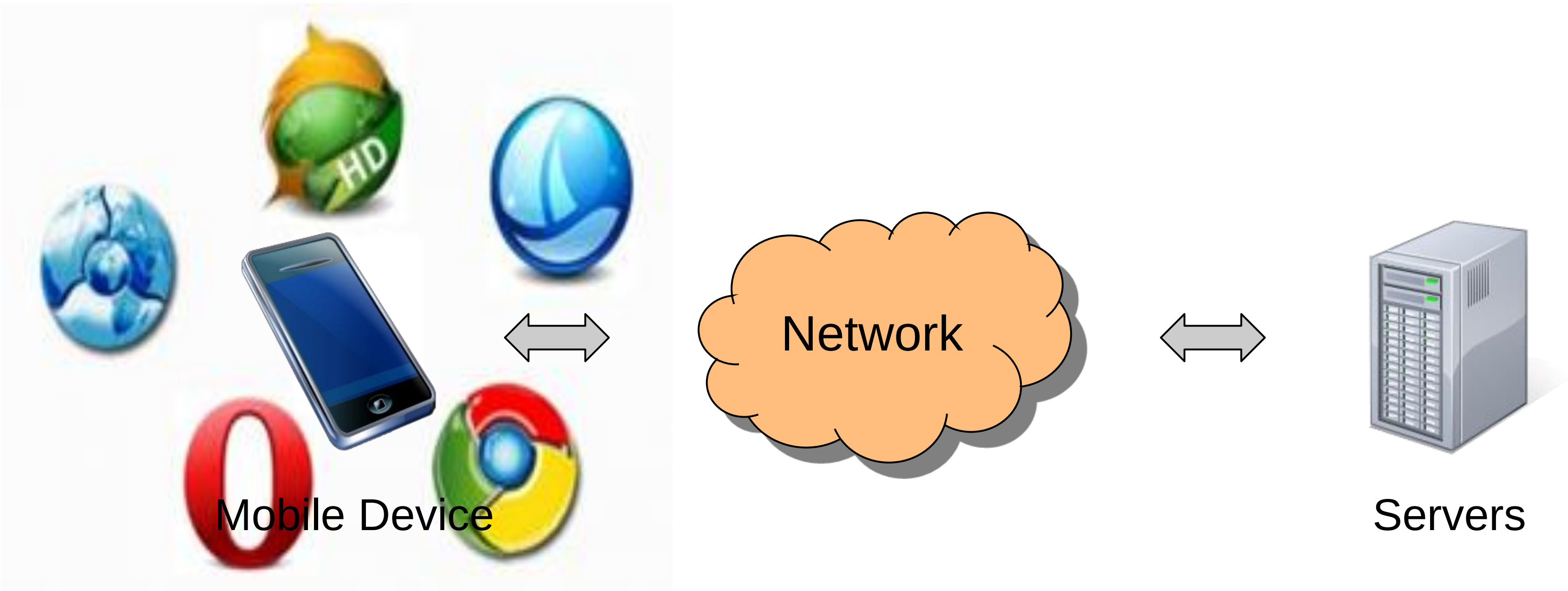
Improving the Energy Consumption of Mobile Web Applications

William G.J. Halfond
Assistant Professor
University of Southern California

Mobile Web Applications



Mobile Web Apps – A Brief History

- First mobile "app" marketplace 2008
  - Now generates over $8.5 billion
- Originally, native apps
  - Choose a side: iOS vs. Android X.X vs. …
  - Bad for developers
  - Bad for users
- HTML 5 and AJAX
  - Better for everybody (except the testers!)

Verification and Validation of Mobile Web Applications

1. Energy consumption
2. User interfaces
3. Software abstractions
4. Security vulnerabilities

Energy Consumption

- Important quality metric
- Strongly influences usability
- Affects perception of app quality
- Goal: Help developers to *understand* and *change* energy consumption in apps
- Current techniques
  - Heavyweight
  - Too course grained
  - Very slow

Areas of Investigation

1. Understand energy consumption in apps
2. Identify best practices and optimizations
3. Automated refactorings and transformations of the implementation

Energy Consumption Projects

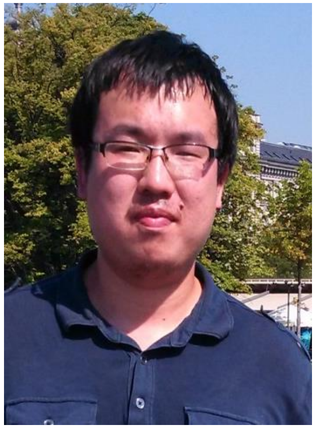
vLens: Map online measurements to source code lines

eLens: Predict energy consumption

Nyx: Automated transformation of web app pages for energy optimization

EDTSO: Energy directed test suite optimization for in-field testing

Shuai Hao



Ding Li

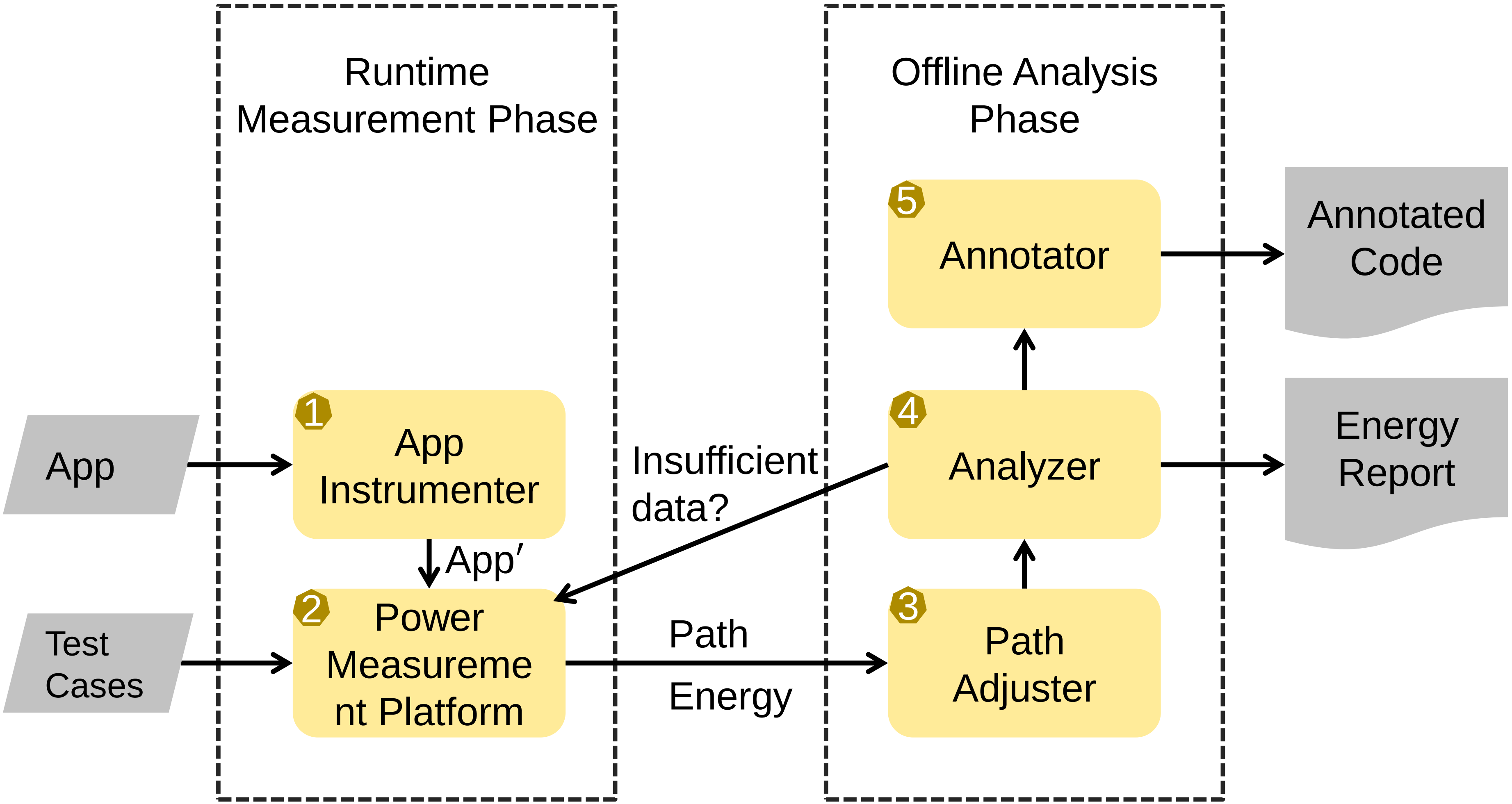
vLens: Online Power Attribution

Goal: Where is the energy being consumed in the application?

Challenges: Insufficient sampling speed, relationships between instructions, minimal runtime overhead

Key insight: Combine static program analysis and statistical analysis techniques

vLens: Process Overview



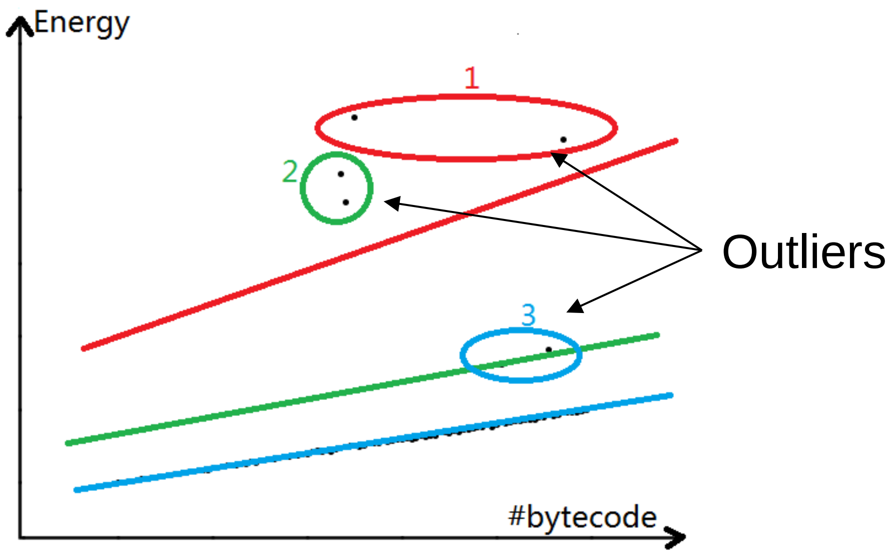
vLens: Statistical Methods

Iterative

$$\sum_i \varphi(y_i - \sum_k x_{ik}\theta_k)x_{ij} = 0$$

$$\varphi_k = \begin{cases} x(k\sigma - x^2)^2, & -k\sigma < x < k\sigma \\ 0, & otherwise \end{cases}$$

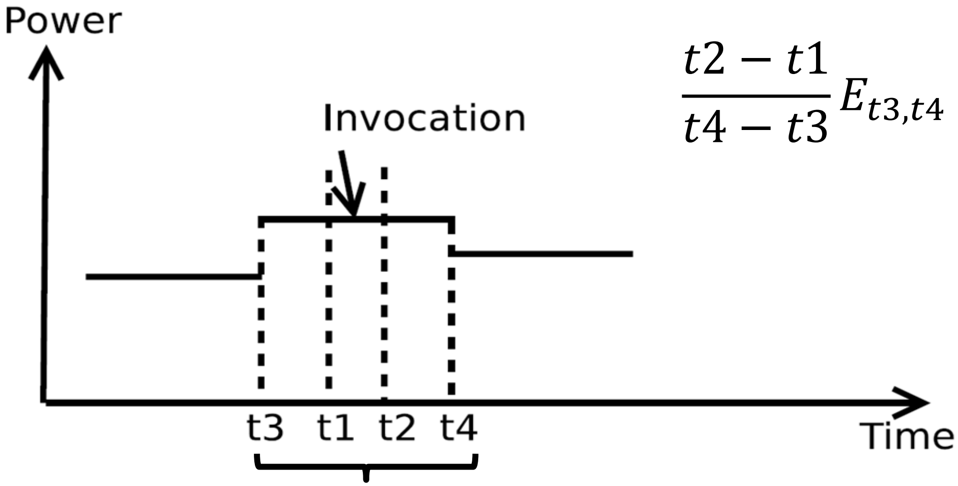
- Handle non-linear API invocations
- Attribute tail energy
- Allocate energy among threads
- Eliminate garbage collection costs
- Remove IPC overhead



Handle Non-Linear Cost Invocations

Calculate and remove non-linear API cost

Sum up energy between entry and exit time stamps



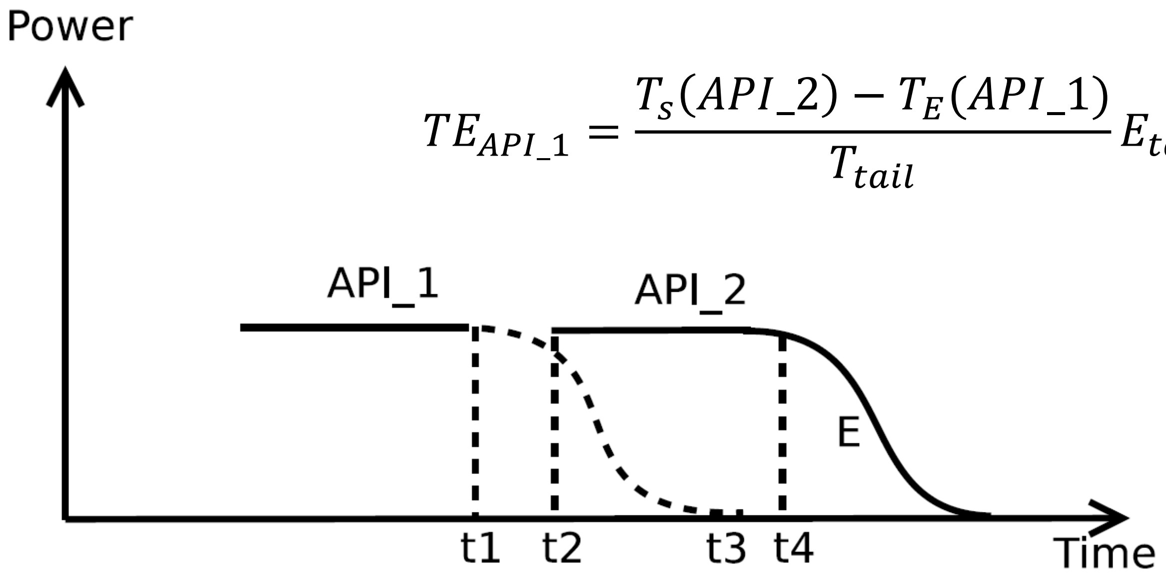
$$\frac{t2 - t1}{t4 - t3} E_{t3,t4}$$

Minimal sampling interval

Attribute Tail Energy

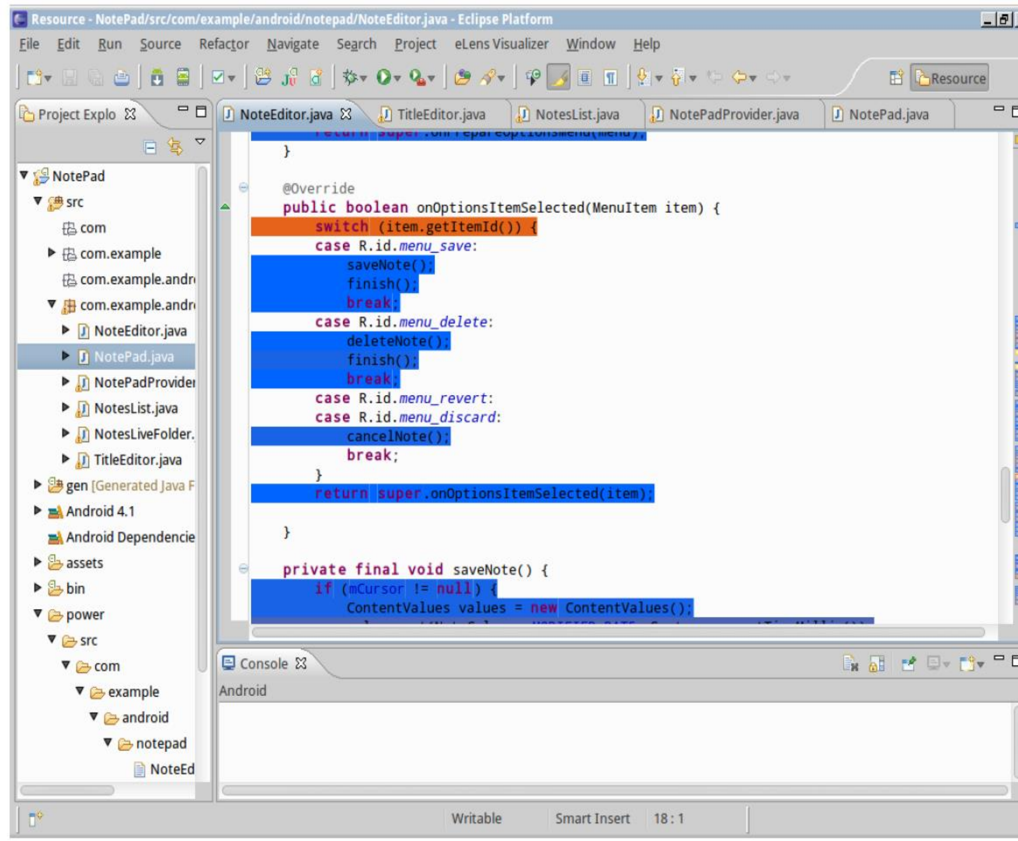
Assign tail energy to corresponding API

$$TE_{API\_1} = \frac{T_S(API\_2) - T_E(API\_1)}{T_{tail}} E_{tail}$$



vLens: Achievements

- High accuracy
  - 9% error
- High granularity
  - On source line level
- Low overhead
  - 4% runtime cost

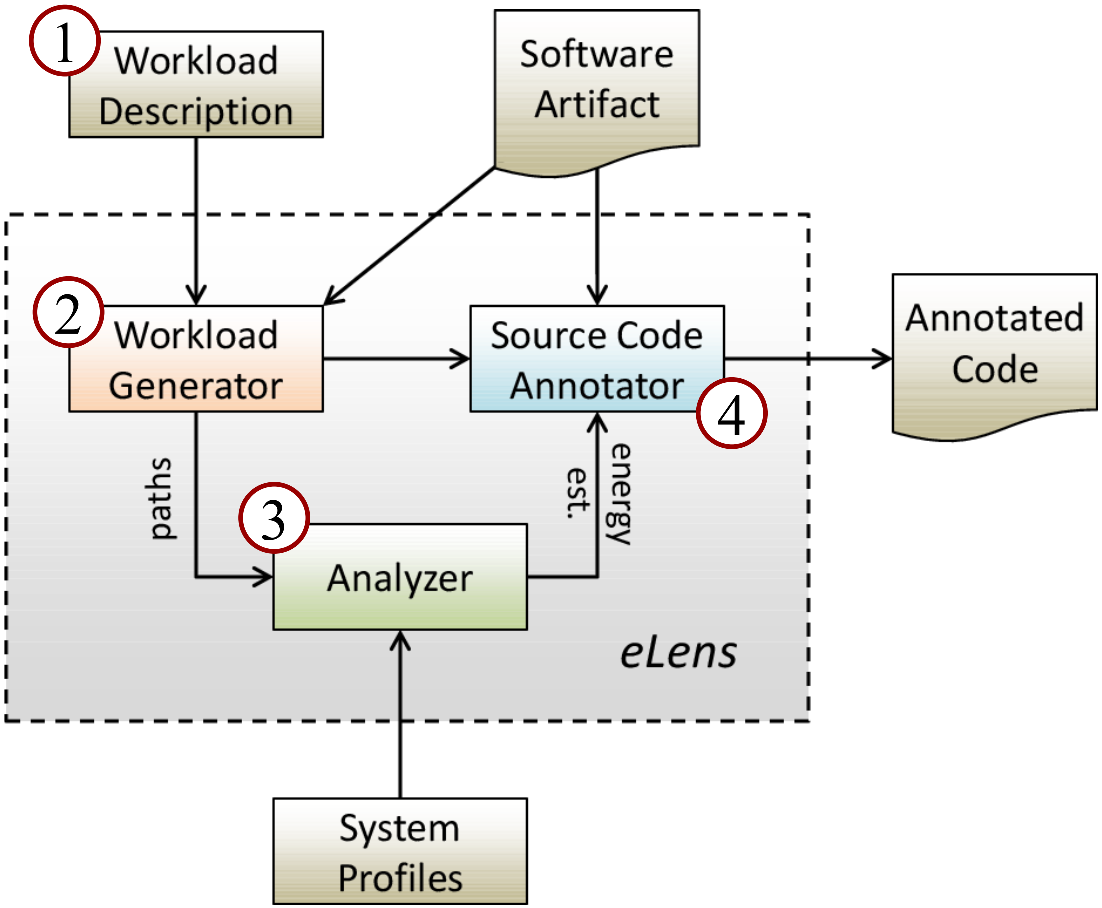


eLens: Predict Energy Consumption

Combine program analysis and per instruction cost modeling

1. Lightweight → no OS changes or specialized hardware required
2. Fine-grained → feedback at the *source* line level
3. Accurate → within 10% of ground truth
4. Fast → estimates within minutes

eLens: Process Overview



1. Generate workload
2. Identify corresponding executed paths
3. Compute power values for paths
4. Annotate source lines

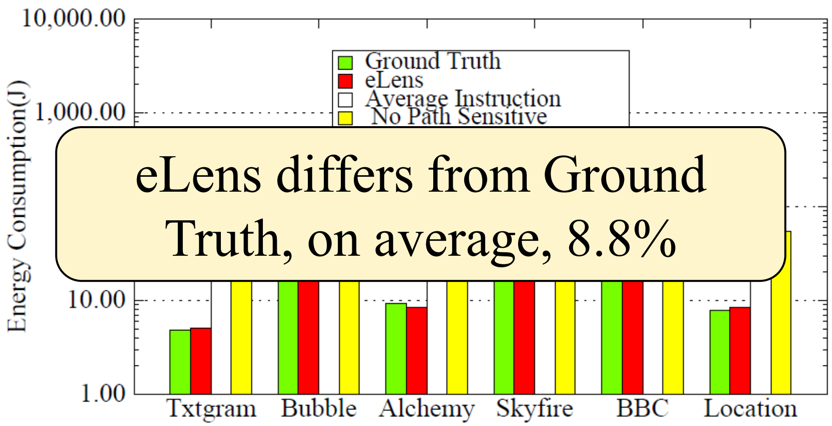
eLens: Energy Calculations

$$\textbf{\textit{Energy}} = \sum_{h \in Hardware} \sum_{i \in path} C_h(i)$$

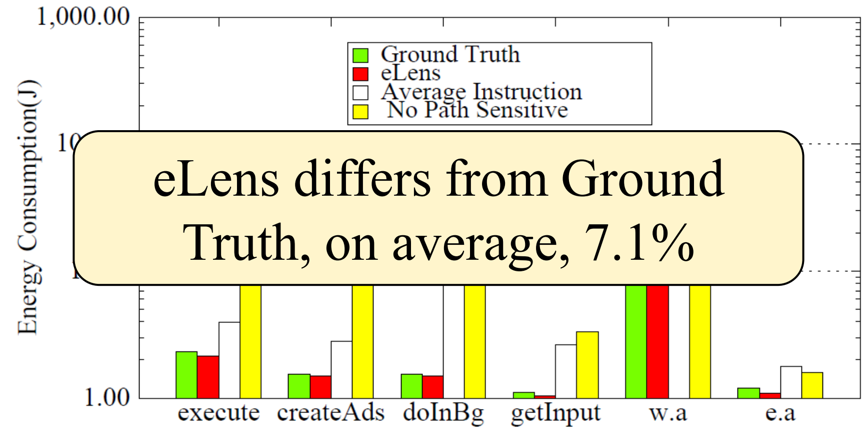
- Cost functions (C_h) for each component (h)
- Instruction's energy cost is either:
  – Path-independent: "fixed-cost" energy
  – Path-dependent: varies based on path

- Cost functions provided by a Software Environment Energy Profile (SEEP)

eLens: Accuracy



Application level



Method level

Application	Error Rate (%)			
	CPU	RAM	WiFi	GPS
BBC Reader	6.2	5.9	-6.8	-
Bubble Blaster II	[illegible]	[illegible]	-11.6	-
Classic Alchemy	[illegible]	[illegible]	-4.4	-
Location	-7.8	-8.4	-	8.1
Skyfire	-7.9	0.9	-8.4	-
Textgram	5.2	4.6	4.6	-

No more than 12% difference from Ground Truth

Research Progression

- vLens and eLens focus on understanding energy consumption
- But we want to optimize!
- What is an energy bug?
  - Informally: Lots of energy is consumed and a more efficient way is possible
- New questions:
  - What are these best practices?
  - How do we automate them?

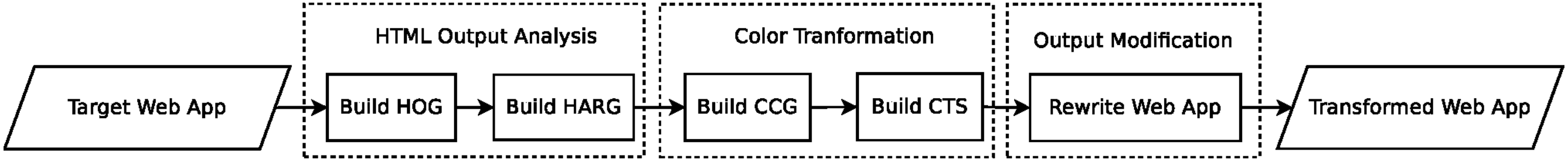
Nyx: Optimizing Mobile Web Apps Display

Problem: Display consumes a significant amount of energy on smartphones

Goal: Automatically transform web applications so that their display consumes less energy

Key insight: Darker colors consume less energy on OLED screens

Nyx: Process



1. HTML Output Analysis
   - Leverage string analysis techniques by Bultan (UCSB) and Moeller (Aarhaus)
2. Color Transformation
   - Model color relationships on page
   - Generate new color mapping scheme
3. Output Modification

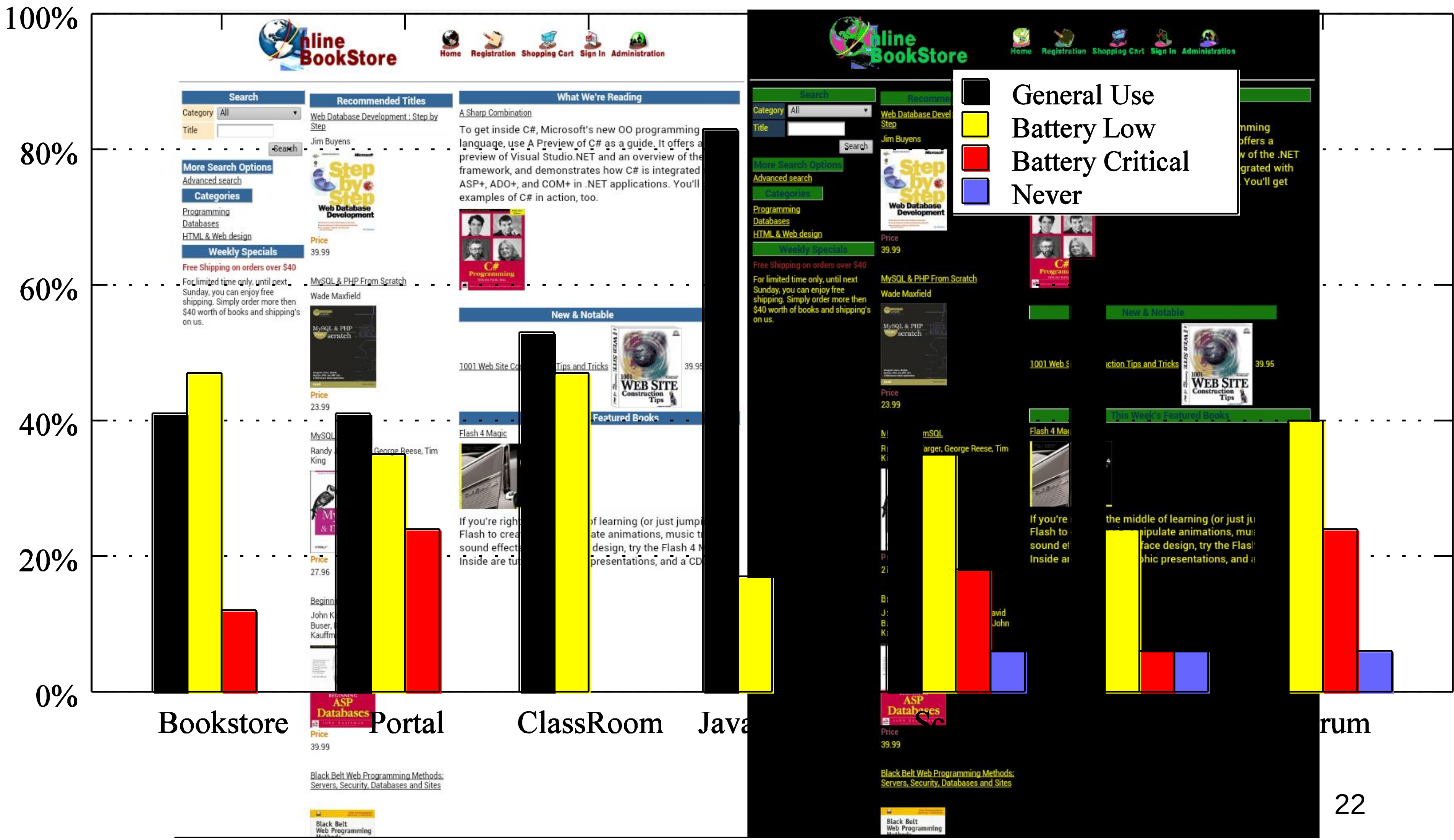
Nyx: Results

App	Loading	Display
Bookstore	26.7	47.2
Portal	24.7	44.2
JavaLi	[illegible]	5.8
ClassR	[illegible]	1.6
Roller	10.4	18.0
Scarab	27.1	47.8
jForum	26.7	47.8

Average 40% savings in display power

Energy savings of transformed applications

Nyx: Transformations



Research Summary

- Verification and validation of mobile web applications
- Energy is an important quality metric
- Step 1: Understand energy consumption
- Step 2: Change energy consumption

Thank you!

halfond@usc.edu