Solar Tab

Version 0.1

Revision History

Name	Date	Changes	Version
Nick Williams	08/02/17	Initial Version	V0.1

Table of Contents

1.0 Overview

This document describes a system that uses SunJoules to match the energy consumption of a single webpage in a browser tab. The working title is SolarTab.

SolarTab uses Javascript and the CloudSolar API to estimate the energy consumption of a discrete viewing activity of a website. The goal of the system is match the entire energy use of the client-side browser tab with solar energy. The system is focused on estimating the energy consumption of the client device and specifically the browser. It does not account for the consumption of other programs or activities being performed by the client device. Likewise, SolarTab does not account for server-side or network transmission energy consumption. The essential goal of this system is to cover the browser tab's energy usage, meaning to over-estimate this energy usage. It is more important for this tool to arrive at a high estimate than it is to arrive at an accurate estimate. Accuracy helps to avoid wasting solar energy, but the main goal is to ensure that enough solar energy is matched to fully account for the true energy usage of the tab.

In this way, SolarTab demonstrates a simple, novel, and useful method for websites to proactively account for the energy consumption they cause on the remote end. Today, some data centers match their energy consumption with solar or other renewable energy, but no easy way to account for the end-to-end energy usage of viewing information remotely is widely available. SolarTab provides a new method to address the client portion of energy usage, but does not at this time attempt to estimate the energy involved in routing and communication of the website across the network from the host to the viewer.

2.0 Approach

The general strategy for calculating the energy usage of a tab with the aim of erring on the high side is to default to conservative estimates and reduce estimates based on information gathered from the client. The system combines three different factors to estimate the tab's energy consumption: the environment, page weight, and page complexity. The first and dominant factor is the environment, which consists primarily of information about the viewing screen and the device type, such as mobile versus desktop or CRT versus OLED. The page weight means the number DOM elements in the page and the length of the page in bytes, considered very roughly proportional to the computational processing required to parse and render the page. The third factor, page complexity, increases the energy consumption of the page by detecting embedded objects, media streams, AJAX, and other interactive elements. Together, these factors roughly account for the electrical cost of a browser tab.

The three factors combine to estimate the power consumption of a browser view, so Javascript can aggregate this usage across time to estimate the energy consumption of a view or session. Note that energy only accumulates when the tab is 'in focus' and visible on the top layer screen in the z-axis. The script updates roughly once per second and adjusts its burn rate locally based on detected changes to the tab. Less frequently, the script contacts CloudSolar to retire more SunJoules and reconfigure.

3.0 Lifecycle

The script generally goes through several phases in the lifecycle of a view or session.[1] First, the script gathers information about the tab environment. Then, it registers itself with CloudSolar and obtains a session ID and a power estimate from CloudSolar. While the script is active, it regularly uses the power estimate to update a SunJoule counter. Every few minutes, it checks in with CloudSolar to obtain more SunJoules. Once the page refreshes, reloads, or closes, the session is over and a new one may begin.

When the script initially registers, CloudSolar uses the information it provides in combination with other indicators such as HTTP headers in the request to provide a power estimate in SunJoules per hour. CloudSolar immediately retires enough SunJoules to account for the first few minutes of the page view and then responds to the script.

In most cases, the script will never contact CloudSolar again because the user has already linked to another webpage or closed the tab. This initial retirement is larger to account for the assumption that much of a typical webpage's energy cost occurs in the initial rendering of the page.

As time passes, the script locally amortizes an energy value based on the power estimate. It updates locally once per second and tracks energy usage in thousandths of SunJoules (on the order of milliwatthours). If the script is still running after the designated time has elapsed, it checks in with CloudSolar and provides its count of elapsed time to indicate that it is still consuming energy. In response, CloudSolar will retire more energy when it receives this 'heartbeat' and may adjust the power estimate if necessary. Note that the elapsed time for the script may be very different from the change in clock time, because the timer stops when the browser tab is out of focus or covered by other interfaces and continues when it is revealed.

CloudSolar uses a database to track these sessions with a lightweight state and there is no need and often no possibility for the script to communicate the end of its lifecycle. A session may be able to match more energy for very long time periods, although most sessions will be extremely short-lived.

4.0 Model

The model for estimating power consumption consists of three primary factors: environment, page weight, page complexity.

4.1 Environment

The general computing environment in which the tab exists provides the majority of the justification for a particular estimate. Two objects are essential to determining the environment, the screen and the platform.

The screen provides informative attributes including aspect ratio, resolution, color depth, and pixel depth. It also indicates the total screen size, the available screen size, and the portion of the screen occupied by this tab at any given time. A screen may account for as much as 60% of a device's

1Note that a session in SolarTab may not be equivalent to other application layer sessions relevant to the browser tab. For example, a webapp may have an authenticated session with a user that persists for more or less than a SolarTab session. A SolarTab session is most closely related to the lifetime of a specific webpage instance.

energy usage[2], so this is the most consequential factor although it is scaled based on the fraction of the screen actually occupied by this tab.

The platform encapsulates other available data about this device. For example, media queries may indicate that a device is very likely a smartphone, in which case the power estimate can be lowered considerably. Even in a larger display like a tablet or notebook, other patterns such as touch support may indicate that a device is newer and more likely to be efficient. Yet other setting values may indicate that the browser or environment has not been updated recently and therefore the most conservative estimates should be used.

The power consumption estimate for the platform is added to the estimate for the screen and then multiplied by the portion of the screen currently taken up by this tab. The result of this calculation is the first and largest factor in the overall estimate accounting for 50-60% of the power estimate.

4.2 Page Weight

Page weight is the second factor, accounting for the remainder of the basis of the overall power estimate. Page weight is calculated by determining the length of the page in bytes and the number of elements in it. Each byte and each element adds a very small wattage to the overall estimate, because a longer page with more elements will take more processing power to load. Page weight is combined with the environmental calculation of the power estimate to form the basis of the overall estimate.

4.3 Page Complexity

Page complexity is the final factor, which effectively adds penalties based on the existence of certain DOM element types including images, embedded objects, audio or visual streams, applets, canvases, linked resources, and other scripts. It is possible for a webpage to have a complexity of zero in which case the other two factors would completely account for the final estimate.

5.0 Limitations

From the sandbox context of Javascript, SolarTab has no way to measure the true power draw of the client device and there are myriad ways in which the use case could make the above model highly inaccurate. These cases should be avoided to the extent possible.

5.1 Unaccounted Views

The script is intended for a single user browsing under typical conditions and cannot necessarily determine every way in which a webpage may be viewed. For example, multiple monitors on one computer or screen sharing with a projector may not result in accurate estimates that cover the entire energy usage to view the page in this instance. Likewise, the energy consumption of programs external to the tab such as screen capture or screen sharing programs are not taken into account. The highest wattage estimates for a given screen and platform are always used, so the estimate should often be several times the actual use of most devices. As long as these situations are atypical, the aggregate estimate for all views of a given website should always exceed the true energy cost.

5.2 High Throughput

SolarTab works best in the context of basic websites, eCommerce sites, and blogs while

2 See Appendix A for references.

websites that exchange large amount of data without refreshing should be cautious when using the SolarTab claim due to major differences from the model's assumptions. Online radio, video streaming sites, and frequently updating webapps are less likely to be accurately estimated by this model, because of how the factors are weighted. In most sites, the initial rendering of the site costs the most energy, but in these types of sites there may be a higher ongoing energy cost than CloudSolar will anticipate.

The third factor, page complexity, is designed to account for this and should do so in most cases, but for webpages that primarily exist as a channel for continually changing content this factor will be more likely to underestimate the true energy cost of this use case for both the monitor and processor. Note also that the SolarTab model does not attempt to account for the energy cost of producing sound, since this is not common in most webpages.

5.3 Manipulation

Since the script will always run on a client device, there is no way to stop knowledgeable clients from manipulating it to provide a result that intentionally exacerbates or mitigates their energy usage. Any modification, emulation, or spoofing of SolarTab voids its claim and may cause that session to be ignored by CloudSolar.

6.0 Security

SolarTab exists to provide an easy way for website administrators to help their viewers support solar energy with minimal risk. No one benefits if the script creates vulnerabilities or exposes user information to other parties, so the script takes certain steps to mitigate such risks:

- All communication with CloudSolar is encrypted using the latest version of SSL or TLS that the client browser supports.
- Although the script obtains various pieces of information about the client environment to feed the model, it does not attempt to gather any information related to the identity of the end user or to tag this user for identification by other scripts or pages. Future versions of the script may optionally request the end user's location to source solar energy from nearby, but the current script does not do anything like this.
- The script does scan the entire webpage, but only to weigh it and identify certain types of elements. The script has no awareness of the human context of a page, the information displayed on a page, or its semantic meaning. It does not take any information from input fields or forms in the page, instead just counting and measuring the page elements.
- In some cases, it may be possible for the SolarTab script to interact with other variables or functions in the page. The aim is to preserve all other page functionality without any impact or naming collision, so please contact us immediately if you suspect that SolarTab is interacting with other parts of a page since this is a bug.
- Domains must register with CloudSolar to use this service. Failure to register with CloudSolar may lead to dropped connections or violations of the same origin policy.
- The script is designed to fail gracefully and to shutdown or do nothing in exceptional circumstances. If anything goes wrong, the solar mark will not appear or will disappear to indicate that solar matching for this view may not be occurring.

If you discover any vulnerabilities related to SolarTab, please contact security@cloudsolar.co.

Appendix A: References

Energy Consumption of Consumer Electronics in U.S. Homes in 2013. Fraunhofer USA: Center for Sustainable Energy Systems. Bryan Urban, Victoria Shmakova, Brian Lim, and Kurt Roth. Revised March 2015.

Review of Computer Energy Consumption and Potential Savings. Megan Bray, 2006.

Power Management Statistics. Northwestern University: Information Technology, 2013.

"How Much Energy Do Computers Use?" by Michael Blue Jay, March 2012.

Appendix B: Terminology

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

CloudSolar – a distribution system for certified solar microcredits

CRT – Cathode Ray Tube

DOM – Document Object Model

HTTP – HypterText Transfer Protocol

ID – Identifier

OLED – Organic Light-Emitting Diode

SSL – Secure Socket Layer

SunJoule – one kiloJoule of solar energy

TLS – Transport Layer Security

XML – eXtensible Markup Language