Chapter 1
Cloud-supported Certification for Energy-Efficient Web Browsing and Services

Gonçalo Avelar, José Simão, and Luís Veiga

Gonçalo Avelar
INESC-ID Lisboa, Universidade de Lisboa - Instituto Superior Técnico

José Simão
INESC-ID Lisboa, Instituto Superior de Engenharia de Lisboa (ISEL/IPL),
e-mail: jsimao@gsd.inesc-id.pt

Luís Veiga
INESC-ID Lisboa, Universidade de Lisboa - Instituto Superior Técnico,
e-mail: luis.veiga@inesc-id.pt

Contents

Abstract Web applications are increasingly pushing more computation to the end-user. With the proliferation of the Software-as-a-Service model, major Cloud providers assume browsers as the user-agent to access their solutions, taking advantage of recent and powerful Web programming client-side technologies. These technologies enhance and revamp web pages aesthetics and interaction mechanics. Unfortunately, they lead to increasing energetic impact, proportional to the rate of appearance of more sophisticated browser mechanisms and web content. This work presents GreenBrowsing, which is composed of i) a Google Chrome extension that manages browser resource usage and, indirectly, energy impact by employing resource limiting mechanisms on browser tabs and; ii) a Certification sub-system, that ranks URL and web domains based on web-page induced energy consumption. We show that GreenBrowsing's mechanisms can achieve substantial resource reduction, in terms of energy-inducing resource metrics like CPU usage, memory usage and variation, up to 80%, for CPU and memory usage. It is also, indirectly and partially, able to reduce bandwidth usage when employing a specific subset of the mechanisms presented. All this with limited degradation of user-experience when compared to browsing the web without the extension.

1.1 Introduction

The Software-as-a-Service business model relies largely on the capacity for the client to execute rich applications inside a Web browser. Parallel to this trend, the Web 2.0 phenomenon also led to the creation of more capable technologies, such as HTML5, enhancements in the JavaScript language and Cascading Style Sheets (CSS), to support blogging platforms, social networks, and multimedia-streaming sites. As a result, the power consumption in a single end-user device, derived from web browsing, is two to three orders of magnitude larger than in all the intermediate routing equipment, found in the traversed network path [14]. This relation between the different machinery that operates on the Internet suggests much more could be done regarding the way web pages are processed and demanded by browsers. To that effect, two scenarios can be considered:

- either people start browsing the web more responsibly, requesting each page at a time, lowering the resource consumption on their devices, and therefore lowering power consumption rates, (which could be perceived as a loss of convenience and business value); or
- developers become more responsible for the software they develop, making energy-efficiency a primary requirement, taking it into account since they start developing their systems.

The first scenario is an improbable one. It is hard to instigate environmental responsibility and energy-awareness into users minds, mainly because the financial and energetic incentives, to make people adopt energy management strategies, are minor compared to the constant "desire for always available computing" [10]. In the same study, it is also suggested that "people do not necessarily choose their automated power management settings". Even though this was a study on energy inefficiencies derived from domestic computer usage, it is reasonable to assume that the same ideas hold in more specific cases like the one of web browsers. Another hint of the users indifference towards green software, can be found in several studies [2, 24], suggesting that energy-awareness must be delegated to the developer, instead of the user.

Therefore, what *power management strategies* should be employed in order to provide power consumption reductions, while browsing the web? How can environmentally concerned users, or even simple-minded users alike, be assured that certain web pages are *greener* than others? How can the related web page processing be used to instigate energy-awareness?

Current solutions lack the context at which they were supposed to perform power management actions (the web browser runtime state). Moreover, they typically oversee components metrics, like CPU utilization, disregarding other important components like main memory, which are also responsible for a reasonable slice of the overall energetic waste [8]. An example is Chameleon [23], that brings power management to the application level, adjusting the speed at which applications run. This might be bad design, since users often impose tight availability constraints on the systems they use. Although techniques such as computation offloading [20] or edge computing servers [5] could be used to improve performance and save energy. these techniques are not always possible to use without hindering the user-experience expected from highly responsive applications.

The main challenge is to provide mechanisms that effectively reduce the energy cost when browsing the web, without sacrificing much of the availability and performance that is expected. This chapter describes and evaluates GreenBrowsing, a system that manages browser access to resources, through the enforcement of different mechanisms that limit resource usage. GreenBrowsing extends the underlying runtime systems and application environments – web browsers – to monitor, promote and certify resource efficiency of running applications – web pages, based on a cloud-supported certification infrastructure.

On the front-end, GreenBrowsing extends Google's Web Browser [1] to reach operating system resource management mechanisms in order to enforce our page-aware energy policies. Chrome embodies a full application execution environment with JavaScript just-in-time compilation, garbage collection, thread and process management, and component-oriented architecture. In essence, a virtual machine for the web. Although it has widespread use, studies also show that it is one of the most power consuming browsers and, in general, one of the most power consuming applications [29, 7]. Supported by a back-end infrastructure running cluster and classification algorithms, GreenBrowsing provides means to certify web pages regarding their energy consumption (both during rending and user-operation), in order to inform users of the energetic inefficiencies related to different web page visualizations. We show that our system significantly saves browsing-related resources (up to 80% for CPU, memory usage and bandwidth usage) while keeping delays almost unnoticeable for the user.

In summary, in this chapter we present the following contributions:

- Policies to manage browser access to resources, through the enforcement of different mechanisms that limit resource usage, by taking into account idle tabs (tabs that are open but not being used).
- An extension to a widely-used browser, Chrome, in order to decrease the energy costs of browsing, as well as taking advantage of the browser API to perform energy-related optimizations.
- A cloud-based energy-related web page certification scheme, based on computational resource consumption, to the end of raising user awareness in regards to what pages are more resource hungry.

The chapter is organized in the following manner. In Section 1.2, both seminal and state-or-the-art energy-reduction systems are surveyed along with energy-related certification systems. In Section 1.3, the architectural choices and the algorithms relative to this work will be described, for both the power management extension and the certification sub-system. Section 1.4 explains the details accounting for platform specific problems and how they were overcome. In Section 1.5, the evaluation methodology will be presented, as well as the evaluation testing done in regards to the resource reduction achieved and user-perceived latency impact. To conclude, Section 1.6 presents final remarks and directions for future work will be given.

1.2 Related Work

In this section we present several mechanisms, techniques and systems related to the area of energy-aware web browsing. Section 1.2.1 focus on techniques to dynamically manage power consumption. Section 1.2.2 presents scheduling algorithms to reduce energy losses, in multi-task environments. Section 1.2.3 discusses big data and energy analytic systems.

1.2.1 Dynamic Power Management

Dynamic Power Management (DPM) is the ability to reduce power dissipation, by selectively turning off, or reducing the performance of a system's components when they are idle (or partially unexploited) [27]. These reductions of power dissipation are typically subject to performance and inherent quality of service constraints.

Benini et al. [6] establish a fundamental approach to system-level dynamic power management by providing an high-level architecture, composed by three main components: the Observer, the Controller and the Policy, (as seen in Figure 1.1). The latter takes power-management decisions. These decisions are based on the information gathered and transmitted by the Observer, as it monitors system activity. The Controller is the component through which power management decisions are enforced, on behalf of the Policy.

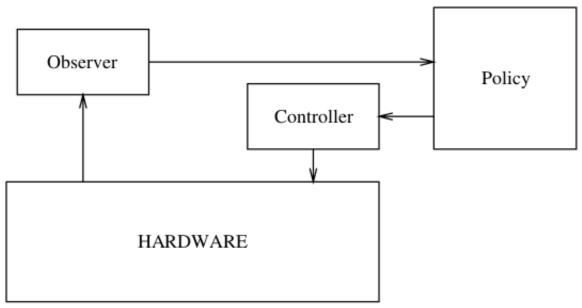


Fig. 1.1 Dynamic Power Management Architecture. As seen in the work of Benini et al. [6]

In practice, the Observer corresponds to the components that interact with the OS and other device APIs, gathering system properties like CPU and memory usage. The controller is the one who engages devices directly through device drives. The Policy is the component responsible for making sense from the gathered data – by the Observer – and issue calls to the right system components – through the Controller.

1.2.1.1 Classification of Dynamic Power Management Systems

The decision criteria that allows for a certain system to be adjusted in terms of power consumption, with respect to a systems state change, is embodied in *Power Models*. Through the enforcement of Power Models, the Policy can adapt to different workload scenarios, adjusting its decision making mechanisms, in order to perform better power management actions. In essence, Power Models provide a formal description of the conditions that need to be met, accounting for both system characteristics and other constraints, (like performance and availability).

Heuristic Power Models.

The more intuitive approach to provide some means of policy adaptation is through the establishment of a static set of rules. These rules are based on common system behaviour and can be implemented as functions, whose parameters correspond to observations and measurements gathered during system's execution. This is the essence of heuristic power models. When modelling simple systems, under near-always-right assumptions, these might suffice in providing good power management capabilities.

Stochastic Power Models.

A stochastic model [19] is one that is based on the notion of stochastic process: set of *random variables X(t)*, as a function of time t, whose values are called states, and the set of possible values is the state space. In this way, a stochastic model models a process where the current system's state depends on previous states in a non-deterministic way.

Amongst the many types of stochastic models, are the widely used Markov Models [17]. In these models the Markov Property [25] holds, hence their name. Intuitively, the Markov Property tells us that given a sequence of N events, the value of the probability of the n^{th} event happening after some exact sequence of $N-1$ previously observed events is approximately equal to the value of the probability of the n^{th} event happening after the $n-1^{\mathrm{th}}$. This approximation is quite useful, since it just requires the computation of the probability of a certain event n^{th}, conditioned to the previous $n-1^{\mathrm{th}}$ one, disregarding all the events observed previously.

In a *controlled system*, the Markov Model state transitions depend on the current state and on an action that is applied to the system. Therefore each state is associated to a certain action. In the context of DPM, it means that when the system is in a certain state, the Policy will perform the corresponding action over some power consuming components. Of course, to that effect, there must be some sort of relation amongst the state set and the components under management, by the Policy.

Typically, the Stochastic Power Models used in Dynamic Power Management fall into the Markov Decision Process category. What Markov Decision Processes (MDP) try to capture is the relation amongst sequences of actions in a system, and the state transitions that they cause.

Learning Power Models.

"An agent is learning if it improves its performance on future tasks after making observations about the world" [32]. This proposition is very relevant

to Dynamic Power Management, because there are some power management problems to which solutions are difficult to be programmed or even devised, due to the complexity of the systems at hand. In this way, the Policy can be conceived as an agent that learns a new Power Model from the data it gathers and actions it performs in run-time. This is why Machine Learning Policies tend to be both Power Model and System Model free, since they learn Power Models dynamically and they might not require any specific system information, in order to execute. They also tend to perform worse than policies that employ Heuristic models, though.

One particular type of learning process is *Reinforced Learning* (RL). In this case, the agent learns from a series of reinforcements: rewards or punishments. No direct consequence of the agent actions is observed, even though some feedback is provided in the form of hints, useful for the agent to reason on how it should operate.

It is often desirable to conceive Dynamic Power Management Policies that perform actions on a trial-and-error basis, learning from good and bad decisions. Hence, they can be designed as Reinforced Learning agents. One common technique of Reinforced Learning is Q-Learning [36] (QL). Q-learning is designed to find stochastic policies, that follow the model of Markov Decision Processes (MDP). This technique is an iterative process with feedback from the previous iterations. At each step of interaction with the environment, the agent observes the environment and issues an action based on the system state. By performing the action, the system moves from one state to another. Based on a value function, the agent decides which action should be taken, given the state the system is in, to achieve the minimum long-term *penalties*. As it is an iterative process, some initial numeral for the value function must be assumed, in order to start the algorithm.

To construct a Markov Decision Processes through Q-Learning, two questions need to be answered: 1) What are the states that compose the state-space? 2) How to formulate cost function, that depends both on the actions taken and states transited to, from the observed information?

System Models.

As shown in the particular cases of Heuristic and Stochastic models, power models often require information regarding the different power states in which systems can be. More precisely, it is often desirable to know how the power state transitions influence performance and the power consumption of systems. To that end, Power Models are often based on *System Models*.

System models are abstract constructs that describe how a system operates and prescribe functionality and interactions amongst different system components. They provide a basic framework of system behaviour, facilitating the conception of suitable power models.

An example of a System Model is the one of Service Requester and Service Provider, (SRSP in short). These systems are composed by four components: a *Power Manager* (PM), a *Service Provider*, (SP), a *Service Requester*, (SR), and a *Service Request Queue*, (SQ). The idea is such that:

- the Service Requester sends requests to the Service Provider;
- the requests are enqueued in the Service Request Queue;
- if the queue of the Service Provider is empty, then it is in idle mode;
- if the queue of the Service Provider is not empty, then it is not in idle mode;
- the PM is able to monitor service requests, and conclude the mode of the Service Provider;

Adaptation.

Power models can be devised statically, before the execution of the policy, or can be dynamically *adapted*, given the history that is maintained, in order to perfect the model, itself.

This is practical because systems workload changes over time, due to the number and type of applications running, users use and misuse of applications and other variable concerns that lead to chaotic and, sometimes, unpredictable power dissipation scenarios. In this way, adapted power models can be employed by policies, changing the criteria by which components are put to sleep or have their performance reduced.

Logically, every policy that employs machine learning techniques to devise its power model is an adaptable policy. Heuristic and Stochastic models can also be adapted in run-time, by any means other than Machine Learning. One limitation of a dynamically generated power model is that it incurs in additional overheads. This is sometimes problematic, especially if the adaptation is computationally intensive or when there are tight performance constraints.

Synchronization.

The way the Policy communicates with the Observer and the Controller is a determining factor on how well the Dynamic Power Manager effectively helps to reduce the power consumption of a system's components. Therefore, it is relevant to classify a Policy regarding its communication *synchrony*, towards the other two DPM components, as synchronous or asynchronous. Typically, asynchronous policies perform better than their counterparts, since they do not incur in overheads as substantial as synchronous policies, by busily waiting for the observer's responses or the controller's actions to succeed. Therefore, they do not miss as many system events that can be relevant to the act of power management and operate in parallel with the Observer and Controller, enhancing performance.

Power Reduction Technique.

Policies can enforce the reduction of power consumption, according to different *technique* types. Either by selectively putting system's components to `sleep` or by *reducing the performance* of those same components. The notion of sleep state will depend of the system that is being managed. One common way of achieving lower power consumption through performance reductions is through *Dynamic Voltage and Frequency Scaling.* DVFS [44, 13] allows the voltage of certain hardware components or the clock frequency of CPUs to be decreased, trading performance for energy. Current architectures provide mechanisms that allow direct access to system components, for DVFS purposes.

Policy Optimality.

Policy classification can be done with respect to *optimality.* Benini et al. [6] also point out that observation is indeed essential for devising good policies, i.e., it is strictly necessary to gather system data and adjust policy decisions in run-time. It is not sufficient to greedily put components to sleep as soon as they are idle. There are trade-offs involved that need to be considered. Namely: 1) in case of multiple sleep states, the Dynamic Power Management System should choose one sleep state over the others and; 2) since transitions to sleep-mode and back to active-mode also have a performance cost and inherent overhead, the DPM System should guarantee that the state transitions actually reduce power, compromising performance just up to an acceptable level. This leads to the problem of *Policy Optimization*, which is the one of choosing a Policy that minimizes power consumption, while under performance constraints, (or vice-versa), based on certain usage patterns. Such a policy is called an *Optimal Policy.*

1.2.1.2 Relevant Dynamic Power Management Solutions.

In the work by Qiu et al. [30], the authors describe the problem of DPM as a continuous-time Markov Decision Process, applied to a SRSP system model. Qiu et al. propose a Continuous Time Process and included the notion of idle and busy states of the Service Provider (SP). This is accomplished by adding a transfer state to the Service Request Queue (SQ), to represent the periods when the SP is busy, (since the SP accesses directly the SQ).

On each iteration, a new policy is generated consisting on the cost of performing a sequence of actions, whose probability is weighted and summed to the delay cost of transiting from one system state to another (the actions could be, for instance to put providers to sleep or wake them up). If the policy is optimal under the performance constraints imposed (an upper-bound to the cost function described), it is put in practice. Otherwise, a new iteration of the algorithm is performed, in order to adjust the sequence of actions that are to be made, and the respective delay costs state transitions.

In the work by Gerards et al. [13], the authors prove in a theoretical fashion that in order to find an optimal schedule for a set of tasks it is necessary to consider both DPM and DVFS, instead of just maximizing idle periods length or minimizing clock frequencies independently. They consider a system model of a number of periodic tasks, in which each of them is invoked the same number of times. The authors conclude that it is best to either start each invocation as soon as possible or as late as possible, being this rationale used to find a globally optimal schedule that minimizes the energy consumption using DPM, for frame-based systems.

In the work by He et al. [15], it is presented a simulated annealing (SA) based heuristic algorithm to minimize the energy consumption of hard real-time systems (real-time system where deadlines must be met) on cluster-based multi-core platforms. It is also proposed a technique that allows the power management algorithm to be executed in an online fashion, exploring the static and dynamic slack (times of idleness, or amount of time left until a new task is scheduled, during job execution).

The system model follows a classic real-time task model, since this solution is intended for multi-core systems. In this way, the system comprehends a task set, where each task corresponds to a pair of its worst case execution time and the deadline (equal to the period of the job the task is executing). The main idea behind SA is to iteratively improve the solution by investigating the neighbour solutions, generated based on penalty and reward values obtained from the solution of the current iteration. If the number of iterations is sufficiently large, an optimal schedule of tasks can be found.

Shen et al. propose an approach [36] to dynamic power management using Reinforced Learning, specifically the *Q-Learning* algorithm. Even though QL can be applied as a model-free technique, the system under management is known before-hand, which allows for the enhancement of the QL algorithm. In this work, they propose a solution to the management of peripheral devices. The policy chosen will consider states that minimize the delay cost at each state and expected average power wasted, given the observations it has made, over the time the algorithm has been executing, while learning from its decisions and maximizing their quality. After a certain set-up time, the optimal policy can be found.

In the work of Wang et al. [43] the authors propose the use of Temporal Difference (TD) learning for Semi-Markov Decision Process (SMDP), as a power model-free technique, to solve the system-level DPM problem. Temporal Difference learning is a type of Reinforcement Learning. The system is modelled as a SRSP model. Temporal Difference Learning assumes that the agent-environment interaction system evolves as a stationary SMDP, which is continuous in time but has a countable number of events. The periods at which those events occur are known as epochs.

The key idea is to separate time in decision epochs. At each decision epoch (corresponding to the SP being in a sleep state) actions are taken, depending on the state of the SR. At the next decision epoch, the action is evaluated in order to associate a value to the action taken previously. This will allow to choose from a set of power preserving actions, for each state of the SR, the one with the most beneficial value. Considering the number of requests from the SR and the total execution time to be fixed, the value function is equivalent to a combination of the average power consumption and per-request latency. The relative weight between average power and per-request latency can be changed, over epochs, to obtain an optimal trade-off curve between the average power and latency per-request.

In Table 1.1 the different algorithms previously presented are summarized according to the classification criteria established in the Section 1.2.1.1. The [-] symbol represents that a certain property is not applicable to a particular solution or that the authors did not specified anything regarding that property.

Work	PowerModel	SystemModel	Policy			
			Optimality	Adaptation	Synchronization	Technique
Qiu et al.	MDP	SRSP	optimal	adaptable	asynchronous	sleep
Gerards et al.	-	Sporadic Tasks	optimal	-	-	DVFS
He et al.	Heuristic	Real-Time Tasks	optimal	adaptable	-	DVFS
Shen et al.	Q-Learning	Peripheral Devices	optimal	adaptable	asynchronous	sleep
Wang et al.	TD Learning	SRSP	optimal	adaptable	-	sleep

Table 1.1 Dynamic Power Management Schemes Classification.

1.2.2 Energy-Aware Scheduling Systems

In the classical definition of scheduling, the goal of the scheduler is to determine which task, thread or process, should be executed, according to some notion of priority. The idea is to optimize and take the most of CPU utilization.

Energy-aware scheduling is the problem of assigning tasks to one or more cores, so that performance and energy objectives are simultaneously met [35]. In this way, the goal of energy-aware scheduling differs from the one of "vanilla" scheduling, since it is intended to solve a multi-objective optimization problem, that comprehends both performance and energy.

Classic techniques include the *First-Come-First-Served* (FCFS) scheduling algorithm [46], where jobs are executed according to the order of their arrival time, to a waiting queue. The major disadvantage of this algorithm is the fact that large jobs greatly delay the execution of the next jobs to execute. This situation is called convoy effect. The *Round Robin* scheduling [40] asserts to each job a time-slice where it can run. Finding the proper value for the time-slices might be challenging to meet performance constraints. Even more if it is intended to achieve mutually performance and power optimization. *Earliest Deadline First* [16] is a dynamic scheduling algorithm where tasks are placed in a priority queue, such that whenever a scheduling event occurs the queue will be searched for the process closest to its deadline, to be scheduled to execution. Because the set of processes that will miss deadlines is largely unpredictable, it is often not a suitable solution to real-time systems.

Energy-aware scheduling impacts on several levels of the system stack. In the work of Kamga et al. [18], they propose a solution where they extend Xen's default Virtual Machine scheduler – *Credit*. The goals are to: i) induce power reduction in the execution of several consolidated VMs while; ii) respecting the agreed Service Level Agreement (SLA) – maintaining acceptable levels of performance. The extension of the Credit scheduler is comprised of two modules: *monitoring module* and *cap control module*. At each tick, the monitoring module gathers the current CPU load for each VM and then computes the optimal frequency to which the CPU should be set to, according to the total VM load and the ratio between current and the maximum frequency. After that, the cap control module re-calculates new cap values for each VM, adjusting each VM CPU share to the fair percentage, taking into account the CPU load of each VM. In this way, it is possible to redistribute unused CPU cycles from one idle or less active VM to another, while minimizing CPU frequency to save energy, respecting the SLAs imposed.

Yan et al. propose an approach [46] where they introduce a job scheduling mechanism that takes the *variation of electricity price* into consideration as a way to make better decisions of the timing of scheduling jobs with diverse power profiles, since electricity price is dynamically changing within a day and High Performance Computing (HPC) jobs have distinct power consumption profiles.

In this approach the scheduling system is composed by three components: a waiting queue, a scheduling window and a scheduling policy. The waiting queue is where jobs are stored in order to be processed by the HPC system. Rather than allocating jobs one by one from the front of the wait queue, the algorithm allocates a window of jobs. The selection of jobs into the window is based on certain user centric metrics, such as job fairness while the allocation of these jobs onto system resources is determined by certain system-centric metrics such as system utilization and energy consumption. By doing so, it is possible to balance different metrics, representing both user satisfaction and system performance.

In the work of Datta et al. [11], the authors present two scheduling algorithms that address the utilization of homogeneous CPUs, operating at different frequencies, in order to lower the global power budget in a multi-processor system. By using *cache miss* and *context switch-CPU migration* indexes, the algorithms are able to exploit the increased performance associated with switching more computationally intensive tasks to higher frequency cores, without suffering from the performance losses associated with cache coherence and context switching overhead. The algorithm assigns static and dynamic priorities to each task. During the schedule stage, the algorithm moves computationally intensive tasks, that perform slower, to a higher frequency core or vice-versa, based on the number of context switches (or cache misses depending on which of the two algorithms is chosen) and their priority.

1.2.3 Energy-related Certification and Analytics on the Cloud

In this section we start by analysing the current solutions that assign some sort of energetic rating to computational systems (Section 1.2.3.1). We then move to the cloud and big data systems domain (Sections 1.2.3.2 and 1.2.3.3) in order to study the relevant work, that will give us insight on how to incorporate an energy-related certification sub-system into GreenBrowsing, following a cloud-based approach.

1.2.3.1 Energy-related Certification Computational Systems.

To our knowledge, there is no considerable work focusing on the energy-related certification of web pages. There is, however, some work that tries to rationalize and quantify the energy consumption of devices and software, for user visualization purposes.

Siebra et al. propose a scheme [37] to certify mobile devices, regarding their energetic performance. The evaluation is done based on mobile operations (voice call, Internet browsing, message services) and temporal delays between them. Each test case has an energy threshold that cannot be surpassed. If it is, then the mobile device under evaluation is not considered to be green. Amsel et al. developed a tool – GreenTracker [3] – that aims at encouraging users to use software systems that are the most environmentally sustainable. They do this by collecting information about the computer's CPU and by comparing software systems in different classes of software (e.g. browsers are compared with other browsers), based on energy consumption. When all the systems in one class have been tested, Green Tracker creates a chart comparing the CPUs across all the software systems.

Camps et al. propose a solution [9] where a classification of web sites depending of their downloadable content is provided to users, making them aware of the web session costs. The classification is done statistically, by computing: i) the average size of objects embedded on pages; ii) the rate flow and; iii) the distance from the web browser to the servers. The energy cost should be displayed to final user: this, from the authors perspective, will allow people to make smarter decisions on how to better manage their energy consumption in their web session.

From these three solutions, the most related to GreenBrowsing is, in fact, the solution presented by Camps et al. However, some disadvantageous characteristics make it less attractive than GreenBrowsing, in particular the fact that it only takes into account the downloadable content of web pages, disregarding important and predominant metrics such as *how heavy the page is* in terms of CPU, memory and I/O performance while rendering and executing JavaScript code. Moreover, all of the required statistical processing is done on the client side of the application, which might turn out to be a dominant overhead, leading to high resource usage and consequent energy consumption.

1.2.3.2 Classes of Big Data Analytics System.

There is a big variability in terms of Big Data systems that deal with energy data. In this section attention will be given to systems that gather home energy counters for auditing, analysis, and automation purposes.

Features.

Singh et al. [39] identifies a number of features that can be used to classify a system, regarding its ability to aggregate data from multiple sources and to ubiquitously control data accesses and sharing (from any device and from anywhere).

- Consolidation: To allow a single view into multiple data streams and cross-correlation between different time series, the system should automatically consolidate energy usage data from multiple sources.
- Durability: To allow analysis of usage history, a consumer's energy data should be always available, irrespective of its time of origin.
- Portability: To prevent lock-in to a single provider, data and computation should be portable to different cloud providers.
- Privacy: To preserve privacy, the system should allow a consumer to determine which other entities can access the data, and at what level of granularity, or employ mechanisms that preserve consumers privacy.
- Flexibility: The system should allow consumers a free choice of analytic algorithms.
- Integrity: The system should ensure that a consumer's energy data have not been tampered with by a third party.
- Scalability: The system should scale to large numbers of consumers and large quantities of time series data.
- Extensibility: It should be possible to add more data sources and analytic algorithms to the system.

- Performance: Data analysis times and access latencies should be minimized.
- Universal Access: Consumers should be able to get real-time access to their data on their Internet-enabled mobile devices.

Design Rationale.

At the highest abstraction level, a system's architecture can be divided into the Data Store (D) components and the Application Runtime (AR) components, that access the data store, and perform the execution of analytic algorithms [39]. If we also consider that the system is comprised by two "endpoints" – one residing locally, at the client-side of the system and other residing remotely – three scenarios for the design of a system are possible:

- *Local-DataStore-Local-Runtime* (LDLR) - Both the Data Store and application Runtime are placed at the client end of the system. There is no remote end.
- *Local-DataStore-Remote-Runtime* (LDRR) - The Data Store is placed at the client side while the application Runtime is executed remotely.
- *Remote-DataStore-Remote-Runtime* (RDRR) - Both the Data Store and application Runtime are placed in the component of the system that operates remotely.

The main disadvantage of the LDLR design is that the application Runtime executes on the client side of the system, which can compromise system performance, due to the computationally intensiveness of the AR execution.

The LDRR design tries to solve the LDLR disadvantage by moving the application runtime to the component of the system that operates remotely. However, as it also happens in the case of the LDLR design, the *Consolidation* feature is harder to attain, since in order to integrate data from various sources into the AR functions, this would incur in greater complexity of the overall system management.

A RDRR design might releases the client-side of the application from the store and application runtime totally, providing a more lightweight approach to the client-end of the system than the LDLR and LDRR designs. However, by moving the Data Store to the remote end of the system, less control over personal data follows, because the granularity at which users can establish access permissions to their energetic data is greatly decreased. This introduces privacy concerns, since certain energy usage patterns might lead to the disclosure of personal habits the users do not intend to make public.

Business Rationale.

This aspect reveals the purpose of the system, which can be classified as a Consumer-Centric system or an Utility-Centric one. The latter emphasizes on the usage of energy data by the system, in order to provide utility planning and operation services such as customer billing and home energy waste visualization [39]. On the other hand, consumer-centric approaches emphasize consumer preferences regarding the way their data are handled [42], by integrating their preferences in the decision-making of the services provided.

1.2.3.3 Relevant Energy-related Big Data Analytics Systems.

In the work of Lachut et al. [21], they present the design of a system for comprehensive home energy measurement with the intent of automating the process of adapting energy demand to meet supply. They do this by measuring how the energy consumption is broken down by each appliance, on house, instead of measuring the overall energetic waste of all appliances or just at individual devices. Instead of having one device measuring the energy consumed by each appliance, which might be considered intrusive, the authors state that only minimal collections of energy-related data need to be gathered, in order to measure the actual energy wasted at each appliance. These devices will provide the necessary metrics in order to statistically determine the energy consumption of each appliance, using a technique based on a Markov Model.

In the work of Lee et al. [22], the authors propose an analytical tool that can assist in assessing, benchmarking, diagnosing, tracking, forecasting, simulating and optimizing the energy consumption in buildings. This tool is deployed in the cloud, in a Software-as-a-Service fashion, performing computationally intensive statistical operations on the data it gathers, and allowing for the visualization of energy-related data of users houses. The visualization is done at costumers devices through a dashboard application that summarizes the data outputted by the tool running in the cloud, alleviating any burden to the customer with regard to software maintenance, ongoing operation and support.

In the work of Singh et al. [39], it is presented a system that allows consumers to control the access to their energy usage data, from different devices on his/her house, and have it analysed on the cloud, using algorithms of their choice. The analysis of their energy-related data can be done by any third party application in a privacy preserving fashion. In order to allow other applications to access to the data stored in the cloud, such as third party applications that can provide different analysis algorithms, privacy protection mechanisms (PPMs) are enforced. This PPMs pre-process data by employing mechanisms like noise addition to the data transfered out of the cloud to these applications.

In the work of Balaji et al. [4], the authors present a system called ZonePAC, for the energy measurement of houses with different types of climatization technologies (e.g. Variable Air Volume type heating, ventilation, air conditioning) and energy consumption feedback provision to the house occupants through a web application. The system makes use of existing sensors present on the deployed physical infrastructure of each building to communicate energy consumption counters from the sensors to a building management web service, called BuildingDepot [45]. It relies on a communication protocol for building automation and control networks, BACnet. The network is formed of sensors and the BACnet Connector that communicate over a BACnet protocol.

In the work of Oliner et al. [26], the authors propose Carat, a system for diagnosing energetic anomalies on mobile devices. This system consists in a client application, running on a client device, to send intermittent, coarse-grained measurements to a server, which correlates energy use with client properties like the running applications, device model, and operating system. The analysis quantifies the error and confidence associated with a diagnosis, suggests actions the user could take to improve battery life, and projects the amount of improvement. The server is deployed in a cloud setting, where the samples from client devices are analysed, aggregating the consumption of various mobile devices.

Table 1.2 presents the features that each system has. [*] means that a partial solution is given. [-] means that the authors give no information regarding that particular feature. Table 1.3 exhibits the classification for each system. To represent the fact that the authors gave no information regarding a specific classification property, the [-] symbol will be used.

Feature	Balaji et al. [4]	Lachut et al. [21]	Lee et al. [22]	Oliner et al. [26]	Singh et al [39]
Consolidation	Yes	No	Yes	Yes	Yes
Durability	-	-	-	Yes	Yes
Portability	-	-	-	Yes	Yes
Privacy	-	Yes	Yes	-	Yes
Flexibility	No	No	No	No	Yes
Integrity	-	-	Yes	-	*
Scalability	-	-	Yes	Yes	Yes
Extensibility	-	-	Yes	-	Yes
Performance	-	Yes	Yes	Yes	Yes
Universal Access	Yes	Yes	Yes	Yes	Yes

Table 1.2 Big Data System Features.

System	Energy Data to Visualize	Design Rationale	Business Rationale
Balaji et al. [4]	Home Energy Consumption	LDLR	Utility-Centric
Lachut et al. [21]	Home Energy Consumption	RDRR	Utility-Centric
Lee et al. [22]	Home Energy Consumption	RDRR	Utility-Centric
Oliner et al. [26]	Mobile Device Energy Anomalies	RDRR	Utility-Centric
Singh et al. [39]	Home Energy Consumption	RDRR	Consumer-Centric

Table 1.3 Big Data System Classification.

1.2.4 Analysis and Discussion

In this section, different energy and software related topics were covered, in order to understand how could a browser power management solution be devised. We presented the trade-offs of Dynamic Power Management, in order to understand the advantages of the different policies presented, as well as help perceiving the most advantageous situations where one could use those different policies. In particular, the concept of Dynamic Power Management Energy-aware scheduling techniques were also discussed because they take into account not only performance constraints but also energetic ones. The rationale of energy-aware scheduling is of great interest to the design of a multi-task architecture. Finally, emphasis was given to the fact that it is desirable to move expensive and resource intensive computations to a cloud-based system when it comes to energy evaluation. These remote systems should be able to process events streams of energy-related counters and give a response in a timely fashion. The data needed to do these computations can sometimes disclose private details of users and so it sould be protected.

1.3 An Architecture for Energy-Efficient Browsing

There are two major subsystems that comprise the GreenBrowsing architecture: a Browser Extension that will act at a power manager, limiting browser access to resources, and a Web Page Certification Back End, to be deployed as a prototypical big data analytics system.

1.3.1 Browser extension and power management

The main roles of the Browser Extension are to reduce the resource consumption of idle tabs, and send to the Analytics back-end resource-related data, used to derive energy consumption data, in order to certify web pages in terms of their energy consumption while being accessed.

A layered view of the extensions proposed are presented in Figure 1.2. *Observer-Controller-Adapter (OCA)* provides interfaces for gathering performance counters of each running tab and the process(es). It is also able to issue commands to reduce tab resource usage through the application of different mechanisms. The *Certification FrontEnd* sends performance data to the back end regarding the open web pages, whose result is rendered by the *Certification Renderer*. The *Policy Enforcer* applies the power reduction algorithms and is configured by the *Policy Manager*. It uses the OCA interface, to gather performance counters and to issue content adaptation and power reduction related commands. The *Web Page Certifier* module will have code to fetch performance counters, through the OCA. It will also interface with the Certification Front End to send the counters gathered to the Back End (for energy-related certification of web pages). Communications with the Certification Renderer are done to inform the user of each web page certification.

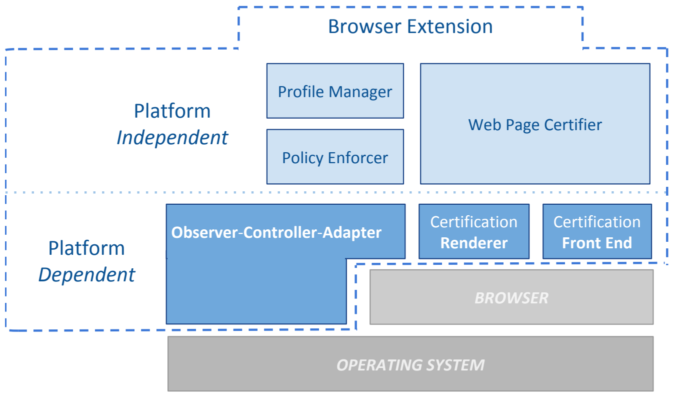


Fig. 1.2 Layered View of The Browser Extension.

In terms of components, the execution of Policy Enforcer's code will be done in parallel with the control and content adaptation of tabs/pages, by two different tasks (comprising one or more threads, each). If they were to be executed sequentially, significant delays could occur in the policy's components execution.

1.3.1.1 Browser-level management policies

We approached the power management problem through simpler heuristics, that offer a smaller implementation overhead compared to stochastic or machine learning techniques. This is particularly important to cause the least possible user-perceived delays, while browsing the web. Two *assumptions* are

```
Data: Windows
Data: Tabs
foreach window in Browser.Windows do
    if window is focused then
        foreach tab in window.Tabs do
            if tab not active then
                compute tab resource usage allowance ;
                apply resource consumption reduction mechanism ;
            else
                give unconditional resource consumption allowance to tab ;
            end
        end
    else
        foreach tab in window.Tabs do
            halt tab's process ;
        end
    end
end
```

Algorithm 1: Tab management algorithm overview.

made, regarding general browsing behaviour, serving as basis to the resource limiting mechanisms to be considered:

- **Last Time Usage.** Tabs that were accessed more recently are more likely to be accessed again and therefore will be less likely to be acted upon. In this way, the tab management policy will make use of a Least Recently Used list for tab energy management.
- **Active Tab Distance.** We also assume that tabs that are closer to the actual tab opened by the user are more likely to be accessed, therefore they will have lesser probability of being discarded or subject to resource constraints.

The pseudo-code at Algorithm 1 summarizes the extension's behaviour for managing idle tab resource consumption. There is an initial test where, if a certain browser window is not focused (i.e. the topmost user-viewed window) all of the processes that handle its tabs will be halted. In other words, they will stop executing. On the other hand, if a certain window is focused, each of its tabs will be acted upon, individually. Firstly, if a certain tab is active (i.e selected by the user) it can consume as many resources it needs. If a tab is not active, the maximum resources its process is allowed to use will be limited. If that tab's process ever reaches the limits imposed, a certain *effect*/action will be cast upon that process. Both the resource type (e.g. CPU usage) and the expected effects on resource limit violation are mechanism-dependent.

An important aspect that our algorithm considers is the maximum resources allowed for a given tab. Equation 1.1 expresses the resource usage factor (uf) which is used to set the maximum resource usage (for any resource type), by taking into account the distance each idle tab is from the currently visualized tab, at a given moment, and the last time a certain idle tab was selected. Considering i as the tab index-distance from a certain tab to the active tab, within a certain window, p as the least-recently-used index relative to tabs within that same window, a as a controllable/user-defined *aggressiveness* exponent to further intensify reductions, if need be, and where $p >= 1$, $i >= 1$, $a >= 0$:

$$uf(i, p, a) = \frac{1}{p \times i^a} \tag{1.1}$$

The value computed by Equation 1.1 will determine the resources that a given *idle tab* can consume, under the influence of any given resource consumption mechanism (e.g. process priority, CPU and memory cap). The intended effect on focused windows' tabs resource consumption is depicted in Figure 1.3.

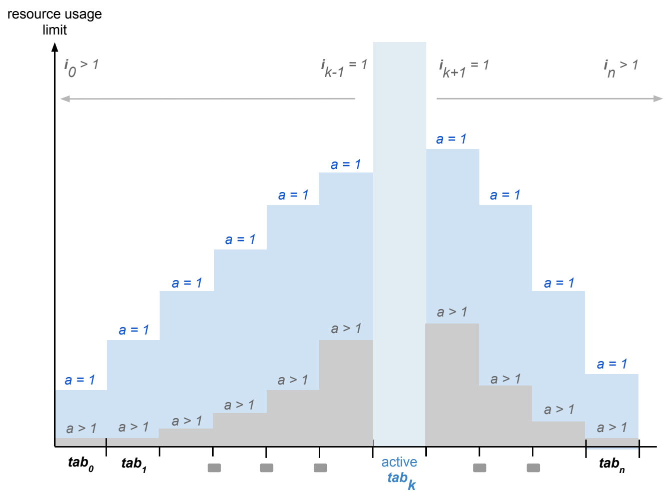


Fig. 1.3 Effects of tab management algorithm on tab resource consumption.

This model allows the possibility that two tabs exist at the same distance i from the active tab and still experience different resource usage limits, for the same value of aggressiveness a, since one of them could have been activated more recently (holding a smaller value for p). One final remark is that some idle tabs may share the same process with the active tab. If this happens, those idle tabs will not be acted upon, since the resource consumption of their process would also constrain the active tab's resource usage (and possibly degrade user experience).

1.3.1.2 Tab management mechanisms

Many browsers employ a multi-process model so GreenBrowsing has to act directly upon the process responsible for handling each tab. This allows to take advantage of some operating system's capabilities, but also implies that some of the mechanisms considered will be OS-dependant. The available GreenBrowsing mechanisms used to reduce resource consumption are presented as follows:

Process Priority Adjustment (prio): If there are x adjustable process scheduling priorities, ascendantly ordered by scheduling weight (where a value of x represents the highest priority value and a value of 1 represents the least), the resulting priority of a certain tab's process will be given by:

$$round(uf(i,p,a) \times x) \tag{1.2}$$

The maximum value x for priority will be the one that represents a standard/normal priority given on process creation, by the operating system scheduler. Regarding the effect on resource limit violation, there is no concrete action taken. The only expectation is for a tab's process to execute less often relative to other processes (browser or any other application's related).

Process CPU Rate Adjustment (cpu): The rate adjustment will be a value in $[0, 100]$, where 0 represents no process usage allowed, and 100 means the process may fully utilize the processor, hence the adjustment will be computed as:

$$round(uf(i,p,a) \times 100) \tag{1.3}$$

If **cpu** is active, once a tab's process CPU usage reaches the limit set for that process, its execution is postponed, running again later, when it is given the chance to do so, by the Operating System's scheduler.

Process Memory Limitation (mem): With this mechanism, the maximum memory allowed for a process will be the maximum committed private memory up to the time that this mechanism was enforced. The adjusted memory value will be given by:

$$round(uf(i,p,a) \times max\_memory\_committed) \tag{1.4}$$

For **mem** there are two versions of this mechanism, with two different possible effect outcomes, once a memory limit is reached by a process: i) a Soft version: the process is halted, and put to a sleep state, returning to execute once its tab becomes active, or; ii) a Hard version: the process is terminated, releasing all the resources allocated by it, until then.

Process Execution Time Limitation (time): In order to limit the duration a certain tab's process is allowed to run for, the average time between consecutive tab activations will be considered. The resource directly managed with this mechanism is execution time. The adjustment formula for allowed process execution time is computed as:

$$round(uf(i,p,a) \times average\_tab\_activation\_time) \tag{1.5}$$

For **time**, the effects employed on limit-breaching processes are the same as with *mem*, once the time for a tab's process to execute expires. It will also wield a Soft and a Hard version.

1.3.1.3 Enforcing limits

Once a certain limit is hit, i.e. the maximum value for a tab to consume was reached or surpassed, the effects on the tab depend on the type of mechanism employed. The effects expected once limits are violated are described as follows:

i) If **prio** is active, there is no concrete action taken, because changing process execution priorities is not, in itself, a resource limiting mechanism. The expected outcome would be, however, for a tab's process to execute less often relative to other processes (browser or any other application's related). But, indeed, the arbitration of when a tab's process should be executed is delegated to the Operating System's scheduler, entirely.
ii) If **cpu** is active, once a tab's process processor usage reaches the limit set for that process, its execution is postponed, running again later, when it is given the chance to do so, by the Operating System's scheduler.
iii) For **mem** there are two versions of this mechanism, with two different possible effect outcomes, once a memory limit is reached by a process:
    - Soft version: the process is either halted, and put to a sleep state, returning to execute once its tab becomes active, or
    - Hard version: the process is terminated, releasing all the resources allocated until then.
iv) For **time**, the effects employed on limit-breaching processes are the same as with *mem*, once the time for a tab's process to execute expires. It will also wield a Soft and a Hard version.

By combining the four resource adjustment metrics with the effects on resource usage limit violation, described previously, a total of *six* mechanisms are singled out. Table 1.4 summarizes these mechanisms in terms of the metric that is directly adjusted by the mechanism, the maximum value for resource limits and action taken on limit violation.

Model	Metric	Maximum Resource Value	Action on limit
prio	cpu usage	Normal process priority	-
cpu	cpu usage	100 % usage	postpone execution
mem soft	memory usage	max memory committed	halt execution
mem hard	memory usage	min memory committed	terminate process
time soft	execution time	avg tab activation time	halt execution
time hard	execution time	avg tab activation time	terminate process

Table 1.4 Mechanisms summarized classification.

1.3.2 Certification Back End

The Certification Back End Sub-System has the objective of *providing a clear and meaningful notion of how much energy web pages consume.* It is composed of three main components, as depited on Figure 1.4:

- A **Certification Server**, comprised of *Network Communication tasks* that receives energy-related web page certification requests and forwards these requests to tasks specialized in the certification of pages themselves (to avoid service bottlenecks and enhancing the scalability of the system regarding the treatment of requests); those are *Analytics Certifier tasks*, that do the work of certifying a given page, according to a specific certification model.
- A **Certification Modeller**, comprised of *Certification Modeller tasks* that adjusts the certification model, having into account all the resource data sent from the extension subsystem. For performance purposes, this design emphasizes the usage of specified *Worker Tasks* to whom parts of the analytical calculations are mapped to. The results of processing data at workers are assembled back at the Modeller Task, as soon as they are ready.
- A **Data Store** that stores the models used in the certification of pages and tuples with information relative to the performance counters of each page;

1.3.2.1 Performance Counters for Energy-related Certification.

The power consumption induced by web pages will be indirectly determined by some of the performance counters gathered on the Browser Extension. For each page, the metrics considered will be:

1. CPU usage (in terms of completed clock cycles);
2. Private (main-)memory usage of processes (in Mega-Bytes);
3. Network interface usage (in terms of the bits-per-second), to process and maintain each page open;

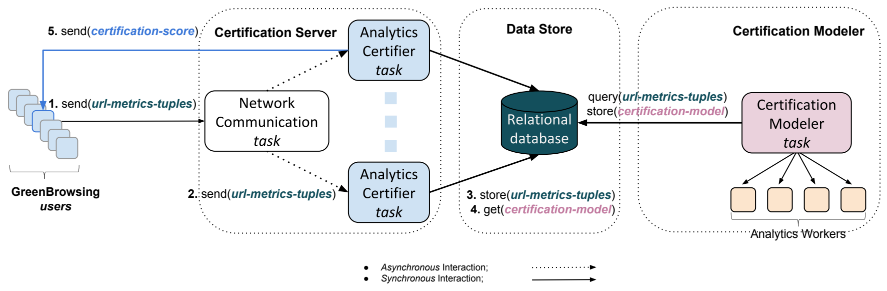

5. send(certification-score)
Certification Server
Analytics Certifier task
Data Store
Certification Modeler
1. send(url-metrics-tuples)
Network Communication task
Relational database
query(url-metrics-tuples)
store(certification-model)
Certification Modeler task
GreenBrowsing users
2. send(url-metrics-tuples)
Analytics Certifier task
3. store(url-metrics-tuples)
4. get(certification-model)
Analytics Workers
Asynchronous Interaction;
Synchronous Interaction;

Fig. 1.4 Certification requests sent from GreenBrowsing users to the Certification Server.

These metrics were chosen because they were proved to be highly related to power consumption, in different settings ([31], [8], [28]).

The certification is done at the level of the *web page* and *domain*, but could easily be extended to individual subdomains, subtrees of each domain hierarchy, for instance. Therefore, the information sent from the Extension to the Certification Back End will be a 5-tuple <*id*, *type*, *CPU-usage*, *memory-usage*, *network-bandwidth-usage*>, where the *type* entry indicates if the performance counters refer to an URL or domain and the *id* refers to its textual representation.

1.3.2.2 Devising Categories and Certifying Pages

Certifying web pages considers the existence of a set of well-defined ranks or certification categories, which in their totality are all-inclusive to any web-page, i.e. given a certain web-page it is always possible to associate a energy-related classification to it. This might not be trivial, since many different resource usage patterns are expected to be observed while processing web-pages, due to the variability of web technologies and richness of web content. This, also because not all resource consumption behaviour inherent to web-page processing is known.

While devising a certification scheme, one should also consider that the entities to certify change over time. Web-pages are no different. What might be considered resource intensive in the present, might be considered acceptable in the future (or, most likely, the other way around). So, in essence, the requirements expected for an appropriate certification scheme, in the context presented, are:

1. Group resource consumption from various sources to ensure all-inclusiveness of certification categories;

2. Predict unobserved resource consumption ranges to further ensure completeness/inclusiveness of certification categories;
3. Dynamically adjust the certification scheme to the changes in web-page properties, that induce varied resource consumption patterns over time;

To devise certification all-inclusive categories from multiple sources the Certification Modeller uses a method know as *Expectation-Maximization* [12]. The basic idea is to cluster the observations recorded into, no less than, 8 categories. This is done in a 3-dimensional (multivariate) random variable space that comprehends one dimension for the CPU usage, one for the memory usage and another for network usage. Two different data sets will be used to compute parameters for two different models – one comprising resource usage associated with URL and another for web-page domains, being the URL dataset contained in the domain dataset.

The observations belonging to the multivariate resource consumption random variables are assumed to be normally distributed, so Multivariate Gaussian Mixture Models ($MGMM$) are used to fit the data and to iteratively train the parameters for 8 random variable's sub-populations, each one corresponding to a cluster. The parameters in question are:

- a 3-dimensional vector comprising the means of each random variable and
- a 3×3 covariance matrix;

After having trained a group of MGMM clusters, a random selection of trained cluster observations is selected from each cluster. The center of mass (CM), or centroid, of each sample is computed, afterwards. The resulting center of mass vector obtained this way, is representative of the category, identifying it unequivocally, and will be used to certify web-page URL or domains while running the certification algorithm. In order to qualify a certain cluster, the vectorial norm of the hypothetical vector space origin to the center of mass of that cluster will be considered. The greater the norm, the more resource intensive pages with that norm's certification category will be considered to be. This is done once, per trained model.

In order to certify a page's URL and domain, tasks running at the Certification Server fetch the clusters' centers of mass, of the last trained Certification Model, from the Data Store. The algorithm to certify a URL/domain's web-page with respect to its consumption consists in comparing the Euclidean distance (d) that goes from each observed resource measurement to the center of mass of each cluster. If two or more clusters' centers of mass are at the same distance from an observation, the one with the greater norm is associated with the observation. In the end, the cluster/category that is associated with more observations, is the final certification category assigned to the URL/domain.

The certification methodology is described more succinctly in Algorithm 2. The input consists of a set of n resource consumption values gathered from a single user device, and a set of k certification categories, previously computed.

```
Input: A set $O = \{O_1, O_2, \ldots, O_n\}$ of resource consumption values
Input: A set $C = \{C_1, C_2, \ldots, C_k\}$ of clusters' centers of mass
Output: A pair $\langle s, k \rangle$, where $s \in \{1, k\}$
$S \leftarrow \{S_1, S_2\}$
for $i \leftarrow 1$ to $n$ do
    $min \leftarrow -\infty$
    $\alpha \leftarrow k$
    for $j \leftarrow 1$ to $k$ do
        $distance \leftarrow d(O_i, C_j)$
        if $distance < min$ then
            $min \leftarrow distance$
            $\alpha \leftarrow j$
        end
    end
    $S_\alpha \leftarrow S_\alpha + 1$
end
$s \leftarrow i$, where $S_i > S_j, \forall \langle S_i, S_j \rangle \in S$
return $\langle s, k \rangle$
```

Algorithm 2: Certification Algorithm used to score web-page URL and domains.

1.4 Browser-level Extensions and Certification Back-End

1.4.1 Browser Extension

The Browser Extension was implemented using a Chrome's deployment on the Windows operating system. Since Chrome has very limited support for process management, namely of its tabs, the Extension needed to be divided in two main entities: (i) **The Browser Extension** itself, comprised of JavaScript callbacks and code rather event-oriented, whose execution and handling is delegated to the Browser, by running from within the Browser itself as a Google Chrome Extension. (ii) A **Background Process (BP)** running natively as a service. Through it, browser processes can be directly managed by communicating, beforehand, with the extension.

The Extension communicates with the Background Process issuing mechanism-related commands and in order to allow the latter to keep track of certain browser state, relevant to the Tab Management Algorithm described at Section 1.3.1.2. The browser state-related information passed this way is composed of general tab information such as tab identifiers, tab indexes within their windows and corresponding process ids. All communications are handled asynchronously by the Background Process each time an event is raised by the browser, following a certain tab state update. For instance when a tab is created, or when a tab is activated.

When on Windows, Chrome uses Windows Job Objects to employ part of its sandboxing constraints. Job Objects are Windows abstractions that allow the grouping of processes and the enforcement of certain limits and restrictions over them. This is exactly what is needed in order to implement the resource limiting mechanisms described at Section 1.3.

The sandboxing used by Chrome prescribes the association of a single tab process to a single Job. Knowing that these Job objects are kept at Chrome's Kernel Process – i.e. the process that orchestrates all browser activity, from tab creation and management to resource access – the BP retrieves these jobs by enumerating all the Windows Kernel Objects present at Chrome's Kernel Process, keeping those that correspond to Job Objects. Once all Job Objects are found, the association of jobs to tab processes is done by calling a Win32 API function. Tab processes are retrieved by querying the browser, through its JavaScript API. This is done at the Browser Extension which, in turn, will pass the tab-to-process associations to the BP, where they are associated with Jobs.

The Tab Management Algorithm described at Section 1.3 will therefore limit resource usage by acting directly on Jobs. Each mechanism is implemented by exploiting the capabilities of Job Objects. For instance, it is possible to change process priorities or adjust maximum CPU rates for any given tab process belonging to a single Job Object. This is accomplished in the cases of *prio* and *cpu* mechanisms.

1.4.2 Certification Back End

Concerning the Back End subsystem, all code was developed on Java. Communication between components is done via the Certification Server Web API, transporting messages in JSON format.

The Certification Server uses the Netty-socketio framework, to serve incoming certification requests. This framework is an implementation of the WebSocket protocol and allows to serve requests efficiently and asynchronously. [1]

The Certification Modeller runs as a process with two Java threads. Each thread computes the model used to certify either URLs or domains. This is done using a combination of Apache Spark built-in Expectation Maximization function, for Multivariate Gaussian Mixtures and Apache Commons Math library, for the sampling of clusters. [2] For storing resource consumption records, coming from the Certification Server, and the model's centers of mass, coming from the Certification Modeller, a PostgresSQL database is deployed at the Data Store. [3]

[1] https://github.com/mrniko/netty-socketio, visited 22 November 2016.

[2] https://spark.apache.org/, visited 22 November 2016.

[3] http://www.postgresql.org/, visited 22 November 2016.

1.5 Evaluation

In order to evaluate GreenBrowsing in a systematic way, tests were scripted combining sequences of *mechanisms* with *aggressiveness* values. The aggressiveness values considered will hold values of 1 and 1024, to assess how the intensification of the limits imposed affects resource usage. A set of typical web-pages was used, comprising pages of news sites, social networks, sports sites, mail clients and multimedia-streaming sites, providing a varied web-page suite.

Scripts were developed to open a set of pages and then navigate through those pages, gathering resource consumption data. Every time a tab is terminated, due to employing *mem hard* or *time hard*, it has its page reloaded once it becomes active again.

Regarding the testing environment Chrome version was 44.0.2391.0, dev-channel release. The operating system on which Chrome was installed was Windows 8.1 Pro – baseline install, no updates. Hardware-wise, the tests were conducted with machines with Intel®Core(TM)2 Duo CPU P8700 running at 2.53GHz, with 4GB of RAM memory.

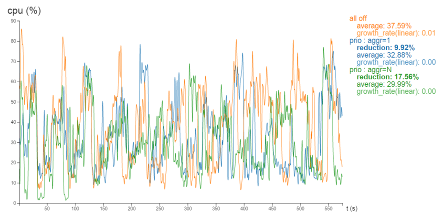
For understanding how the employment of certain mechanism combinations might affect Latency, browsing habits are simulated through different *tab selection policies*. These policies state what is the next tab to activate (i.e. what page to visualize next): i) *round-robin selection* to navigate sequentially from tab to tab; ii) *central tab incidence*, where the tabs at the center of the tab bar will be selected more often, by following a periodic navigation scheme, from the first tab to the last and from the last to the first one, in a back and forth-fashion; iii) *random tab selection* where a certain tab is selected randomly, possibly more than once.

1.5.1 Resource Usage Evaluation

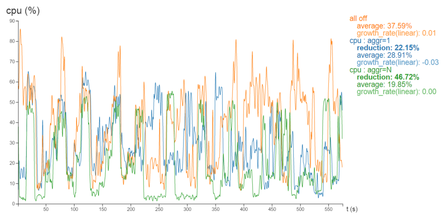
The resource variations induced by *prio* might not noticeable do to the naked eye because of the highly variable values of CPU usage rates, over time. Reductions of 9.92% and 17.56% were recorded, however, being the latter recorded with an higher value of aggressiveness, as shown in Figure 1.5, in green.

When applying *cpu* (Figure 1.5), the reductions in CPU usage are intensified even more when compared with *prio*, this time holding reductions that range from 20% to about 47% of CPU time. This seemingly advantage over *prio* was expected, since *cpu* directly adjusts the CPU usage allowed for each tab's process, contrary to *prio*, that associates priorities to a process without adjusting the maximum value for CPU usage, itself.

Figure 1.6 depicts how applying *mem soft* and *time soft* influenced CPU usage. The first seems to be the most prominent in reducing CPU usage, with

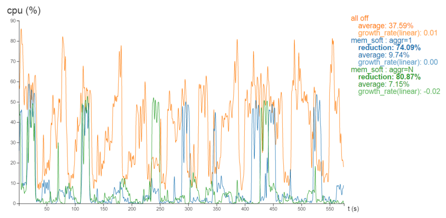


(a) CPU usage for *prio*.

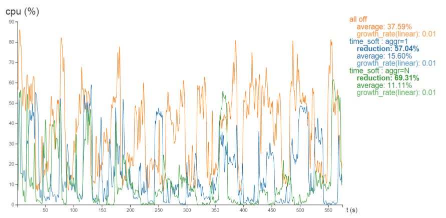


(b) CPU usage for *prio* & *cpu*.

Fig. 1.5 Priority and CPU share mechanisms

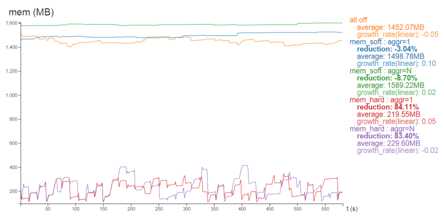


(a) CPU usage for *mem soft*.

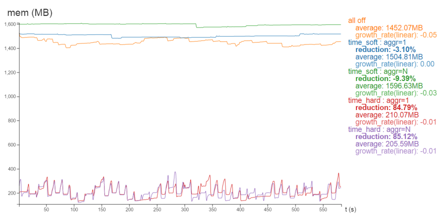


(b) CPU usage for *time soft*.

Fig. 1.6 CPU usage when applying memory related mechanisms



(a) Memory usage for *mem soft* & *mem hard*.



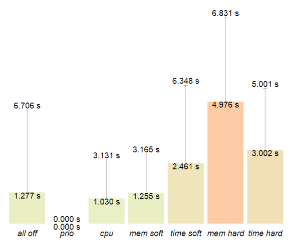
(b) Memory usage for *time soft* & *time hard*.

Fig. 1.7 Memory restriction mechanisms

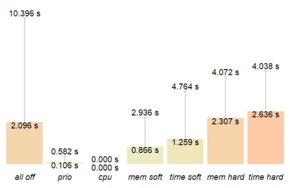
80% reductions, while the latter is still successful in doing so, even though to a lesser extent, achieving close to 70% reductions.

Concerning memory usage, depicted in Figure 1.7, *hard* mechanisms induce a substantially lower memory usage, than their *soft* counterparts, achieving reductions of 80% to 85%, when compared to mechanisms being *all off*.

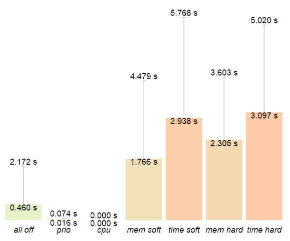
Overall, *mem soft* and *time soft* seemed to be the most capable mechanisms, in terms of managing idle tab resource consumption regarding CPU usage. Even though experiments in Figure 1.7 seem to disprove its effectiveness in reducing memory usage, (since *soft* mechanisms achieved slight increases when compared to *all off*), it is important to notice how stable memory consumption was when compared to the memory variations induced by other mechanisms and *all off*, over time. If it is assumed that memory



(a) Latency measurement / round-robin tab selection



(b) Latency measurement / central-tab-incidence selection



(c) Latency measurement / random tab selection

Fig. 1.8 Latency measurements for the 3 tab selection policies considered.

variations represent system-wide activity, due to having many system entities accessing it, and therefore inducing energy consumption rates proportional to the variations recorded, then *soft* mechanisms effectively help reduce energy consumption, by varying the least.

1.5.2 Perceived Delays Evaluation

In order to assess what are the implications in terms of user experience-significant requirements, Latency was recorded, while running resource consumption tests. Latency, in this context, corresponds to the time period that goes from the moment the active tab starts loading web-page content to the moment that content is totally loaded. This notion of latency is useful to give an idea of how much time is wasted, by enforcing certain mechanisms, in comparison to others.

Figure 1.8 presents the latencies experienced on average, as rectangles, for each tab selection policy. Standard deviations correspond to the vertical lines above rectangles. It is possible to see that latencies for *hard* mechanisms were always bigger, on average, when compared to other mechanisms. The experiments comprising *all off*, *prio* and *cpu* held the smaller latency values, as expected, since they tamper very little with process functioning, when compared to other mechanisms (namely the *soft* and *hard* ones). It is possible to observe that *soft* mechanisms seem to achieve acceptable latencies, when compared to *all off*. The exception is when tabs were chosen randomly, where the latency values are comparable to those recorded for *hard* processes. The standard deviations observed are rather high in value. It has to do with the wide latency-value-ranges recorded since, occasionally, some long periods of consecutive busy-tab activations were recorded (where the activated tabs were still processing their pages).

It seems, therefore, negotiable to apply all mechanisms for resource reduction purposes, with the exception of *hard* mechanisms, given the latencies recorded for them, in most experiments.

1.6 Conclusions

This chapter presented GreenBrowsing, a tab-management solution (implemented as a Google Chrome extension) and a cloud-based energy-related certification scheme implemented on a separate sub-system. Evaluation shows *substantial resource usage reductions*, on energy consumption-related resource metrics (up to 80% for CPU, 85% for memory usage and 85% for bandwidth usage) while preserving acceptable user-perceived delays (unnoticeable in most cases). All of this when comparing GreenBrowsing-aided web-navigations with standard navigations.

Regarding future work, more resource reduction mechanisms could be devised in order to account for bandwidth usage, since studies show it plays a significant part on energy consumption, specially in the case of Wi-Fi enabled devices. The Back End would benefit from improvements at the Data Store, in order to improve its scalability when it comes to processing reads and writes of resource consumption records. Furthermore, we would like to explore how previous work on differentiated quality-of-service in the cloud [38] could be combined with declarative policies [41] in order to improve the approach effectiveness and expressiveness for users. Also relevant is studying how this work can be combined with providing web based services from community networks in order to further improve energy effectiveness [34, 33].

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia with reference PTDC/EEI-SCR/6945/2014, and by the ERDF through COMPETE 2020 Programme, within project POCI-01-0145-FEDER-016883, the Engineering School of the Polytechnic Institute of Lisbon (ISEL/IPL).

References

1. Chrome browser. https://www.google.com/intl/en/chrome/browser/
2. Amsel, N., Ibrahim, Z., Malik, A., Tomlinson, B.: Toward sustainable software engineering (nier track). In: Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pp. 976–979. ACM, New York, NY, USA (2011)

3. Amsel, N., Tomlinson, B.: Green tracker: A tool for estimating the energy consumption of software. In: CHI '10 Extended Abstracts on Human Factors in Computing Systems, CHI EA '10. ACM, New York, NY, USA (2010)
4. Balaji, B., Teraoka, H., Gupta, R., Agarwal, Y.: Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In: Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, BuildSys'13. ACM, New York, NY, USA (2013)
5. Beck, M.T., Werner, M., Feld, S., Schimper, T.: Mobile edge computing: A taxonomy. In: The Sixth International Conference on Advances in Future Internet (2014)
6. Benini, L., Bogliolo, A., Cavallucci, S., Riccó, B.: Monitoring system activity for os-directed dynamic power management. In: Proceedings of the 1998 International Symposium on Low Power Electronics and Design, ISLPED '98. ACM, New York, NY, USA (1998)
7. Bianzino, A.P., Raju, A.K., Rossi, D.: Greening the internet: Measuring web power consumption. IT Professional **13** (2011)
8. Bircher, W.L., John, L.K.: Complete system power estimation using processor performance events. IEEE Transactions on Computers **61**(4), 563–577 (2012)
9. Camps, F.: Web browser energy consumption (2010)
10. Chetty, M., Brush, A.B., Meyers, B.R., Johns, P.: It's not easy being green: Understanding home computer power management. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09. ACM (2009)
11. Datta, A.K., Patel, R.: Cpu scheduling for power/energy management on multicore processors using cache miss and context switch data. IEEE Transactions on Parallel and Distributed Systems (2013)
12. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of The Royal Statistical Society, Series B **39**(1), 1–38 (1977)
13. Gerards, M., Kuper, J.: Optimal dpm and dvfs for frame-based real-time systems. TACO **9**(4) (2013)
14. Gyarmati, L., Trinh, T.A.: Power footprint of internet services. In: Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking, e-Energy '11. ACM (2011)
15. He, D., Mueller, W.: A heuristic energy-aware approach for hard real-time systems on multi-core platforms. In: Proceedings of the 2012 15th Euromicro Conference on Digital System Design, DSD '12, pp. 288–295. IEEE Computer Society, Washington, DC, USA (2012)
16. Jansen, P.G., Mullender, S.J., Havinga, P.J., Scholten, H.: Lightweight edf scheduling with deadline inheritance. Tech. rep., University of Twente. May (2003)
17. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1-2) (1998)
18. Kamga, C.M., Tran, G.S., Broto, L.: Extended scheduler for efficient frequency scaling in virtualized systems. SIGOPS Oper. Syst. Rev. **46**(2) (2012)
19. Klebaner, F.C.: Introduction to stochastic calculus with application (3rd edition) (2012)
20. Kumar, K., Liu, J., Lu, Y.H., Bhargava, B.: A survey of computation offloading for mobile systems. Mobile Networks and Applications **18**(1), 129–140 (2013). DOI 10.1007/s11036-012-0368-0. URL http://dx.doi.org/10.1007/s11036-012-0368-0
21. Lachut, D., Piel, S., Choudhury, L., Xiong, Y., Rollins, S., Moran, K., Banerjee, N.: Minimizing intrusiveness in home energy measurement. In: Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, BuildSys '12. ACM, New York, NY, USA (2012)
22. Lee, Y.M., An, L., Liu, F., Horesh, R., Chae, Y.T., Zhang, R., Meliksetian, E., Chowdhary, P., Nevill, P., Snowdon, J.L.: Building energy performance analytics on cloud as a service. Serv. Sci. (2013)

23. Liu, X., Shenoy, P., Corner, M.: Chameleon: Application level power management with performance isolation. In: Proceedings of the 13th Annual ACM International Conference on Multimedia, MULTIMEDIA '05. ACM, New York, NY, USA (2005)
24. Miettinen, A.P., Nurminen, J.K.: Analysis of the energy consumption of javascript based mobile web applications. In: MOBILIGHT (2010)
25. Norris, J.R.: Markov chains. Cambridge Series in Statistical and Probabilistic Mathematics (1998)
26. Oliner, A.J., Iyer, A.P., Stoica, I., Lagerspetz, E., Tarkoma, S.: Carat: Collaborative energy diagnosis for mobile devices. SenSys '13. ACM, New York, NY, USA (2013)
27. Paleologo, B.B., Benini, L., Bogliolo, A., Paleologo, G.A., Micheli, G.D.: Policy optimization for dynamic power management. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **18**, 813–833 (1998)
28. Park, J., Yoo, S., Lee, S., Park, C.: Power modeling of solid state disk for dynamic power management policy design in embedded systems. In: Proceedings of the 7th IFIP WG 10.2 International Workshop on Software Technologies for Embedded and Ubiquitous Systems, SEUS '09, pp. 24–35. Springer-Verlag, Berlin, Heidelberg (2009)
29. Patel, S., Perkinson, J.: Fraunhofer report - the impact of internet browsers on computer energy consumption (2013)
30. Qiu, Q., Pedram, M.: Dynamic power management based on continuous-time markov decision processes. In: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference, DAC '99. ACM, New York, NY, USA (1999)
31. Rodrigues R. Koren, I.K.S.: A study on the use of performance counters to estimate power in microprocessors. In: Circuits and Systems II: Express Briefs, IEEE Transactions (2013)
32. Russel, S., Norvig, P.: Artificial intelligence: A modern approach (3rd edition) (2009)
33. Sharifi, L., Cerdà-Alabern, L., Freitag, F., Veiga, L.: Energy efficient cloud service provisioning: Keeping data center granularity in perspective. J. Grid Comput. **14**(2), 299–325 (2016). DOI 10.1007/s10723-015-9358-3. URL http://dx.doi.org/10.1007/s10723-015-9358-3
34. Sharifi, L., Rameshan, N., Freitag, F., Veiga, L.: Energy efficiency dilemma: P2p-cloud vs. datacenter. In: IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014, Singapore, December 15-18, 2014, pp. 611–619. IEEE Computer Society (2014). DOI 10.1109/CloudCom.2014.137. URL http://dx.doi.org/10.1109/CloudCom.2014.137
35. Sheikh, H.F., Tan, H., Ahmad, I., Ranka, S., Bv, P.: Energy- and performance-aware scheduling of tasks on parallel and distributed systems. J. Emerg. Technol. Comput. Syst. **8**(4) (2012)
36. Shen, H., Tan, Y., Lu, J., Wu, Q., Qiu, Q.: Achieving autonomous power management using reinforcement learning. ACM Trans. Des. Autom. Electron. Syst. **18**(2) (2013)
37. de Siebra, C., Costa, P., Marques, R., Santos, A.L.M., da Silva, F.Q.B.: Towards a green mobile development and certification. IEEE (2011)
38. Simão, J., Veiga, L.: Flexible slas in the cloud with a partial utility-driven scheduling architecture. In: IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 1, pp. 274–281. IEEE Computer Society (2013). DOI 10.1109/CloudCom.2013.43. URL http://dx.doi.org/10.1109/CloudCom.2013.43
39. Singh, R.P., Keshav, S., Brecht, T.: A cloud-based consumer-centric architecture for energy data analytics. In: Proceedings of the Fourth International Conference on Future Energy Systems, e-Energy '13. ACM, New York, NY, USA (2013)
40. Tanenbaum, A.S.: Modern Operating Systems, 3rd edn. Prentice Hall Press, Upper Saddle River, NJ, USA (2007)
41. Veiga, L., Ferreira, P.: Poliper: policies for mobile and pervasive environments. In: F. Kon, F.M. Costa, N. Wang, R. Cerqueira (eds.) Proceedings of the 3rd Workshop on Adaptive and Reflective Middleware, ARM 2003, Toronto, Ontario, Canada,

October 19, 2004, pp. 238–243. ACM (2004). DOI 10.1145/1028613.1028623. URL http://doi.acm.org/10.1145/1028613.1028623
42. W. Liu, K.L., Pearson, D.: Consumer-centric smart grid. Innovative Smart Grid Technologies pp. 1–6 (2011)
43. Wang, Y., Xie, Q., Ammari, A., Pedram, M.: Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification. Proceedings of the 48th Design Automation Conference on - DAC '11 p. 41 (2011)
44. Weiser, M., Welch, B., Demers, A., Shenker, S.: Scheduling for Reduced CPU Energy (1994)
45. Y. Agarwal R. Gupta, D.K., Weng, T.: Buildingdepot: An extensible and distributed architecture for building data storage, access and sharing. In proc. of the 4th ACM Workshop on BuildSys. ACM (2012)
46. Yang, X., Zhou, Z., Wallace, S., Lan, Z., Tang, W., Coghlan, S., Papka, M.E.: Integrating dynamic pricing of electricity into energy aware scheduling for hpc systems. In: Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, SC '13. ACM, New York, NY, USA (2013)