GreenBrowsing

Gonçalo João Curado Avelar
goncalo.avelar@ist.utl.pt

Instituto Superior Técnico
INESC-ID

Abstract. Web 2.0 allowed for the enhancement and revamp of web pages aesthetics and interaction mechanics. Moreover, current web browsers function almost as a *de facto* operating system: they run "apps", along with other background plug-ins. All of which has an increasing energetic impact, proportional to the rate of appearance of more sophisticated browser mechanisms and web content. To that end, we present the architecture of GreenBrowsing. A system that proposes the provision of (i) a Google Chrome extension to monitor, rationalize and reduce the energy consumption of the browsing experience and (ii) a Certification Scheme for dynamic web pages, based on web-page performance counter statistics and analysis, performed on the cloud.

Keywords: web browser, web page certification, green IT, energy efficiency, power consumption

1 Introduction

As computing systems evolve, the energy spent in the provision of IT services increases. The carbon footprint of IT machinery becomes more evident and the energy costs of IT keep rising. As of 2008, the estimate of greenhouse gas emissions resultant from computer usage, was approximately a ton of carbon dioxide every year [29]. The trend is that the volume of emissions continues to grow.

In order to create more sustainable and energy efficient computing systems, measures must be taken regarding the ways systems are designed, used, manufactured and even disposed [29]. This applies to servers, networking infrastructure components, (like switches and routers), and the devices the end-users resort to.

The means to reduce power consumption could be both hardware or software based. However, in the context of the World Wide Web and at the scope of end-user devices, the web browser should be one of the components to focus on, when it comes to power management and tunning.

The improvements in connectivity and delivery of content in the last few years, made it possible to share a lot more information than it, otherwise, would. The Web 2.0 phenomenon lead to the creation of more capable technologies, (HTML5, CSS, JavaScript), powering blogging platforms, social networks, and multimedia-streaming sites. By being assembled with these technologies, website contents are sent to web browsers where they are processed and, more often, behave like a true application than a static web page. 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 [18]. This relation, between the different machinery that operate over the Internet, suggests that 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);
- developers become more responsible for the software they develop, making energy-efficiency a primary requirement, by taking it into account from the start of the development cycle of 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" [13]. Another hint of the users indifference towards green software, can be found in a study by Amsel et al. [6]. It seems that energy-awareness must be delegated to the developer, instead of the user. In fact, some already argued in favor of that [28]. The problem is that it is not feasible to reconstruct existing web applications, in particular because of the programmatic effort it requires.

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

These questions motivate the solution we propose, as an alternative to the scenarios discussed previously, by extending the underlying runtime systems and application environments – web browsers – to monitor, promote and certify resource efficiency of running applications – web pages.

The main challenge of this work 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, while browsing, and by providing means to certify web pages energetically-wise, in order to inform users of the energetic inefficiencies related to different web page visualizations.

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 [11]. An example is Chameleon [26], 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. Reducing application run speed might lead to negative user experience. On the other hand, there is ACE [51]. This systems tries to leverage the web browsing networking behavior, by reducing user-perceived page fetch latency, allowing also for energetic gains. However it focus solely on power management details of connections, disregarding other resource hungry components of a web browser, like the idle pages.

This work will focus specifically in the Google Chrome Web Browser [2]. Chrome is a complete web browser. By complete it is meant to be more than hyper-text page retriever. It 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. Chrome also allows for the installing of extensions, that range from games to plug-ins with daemon-like behavior, (e.g. Addblock). As a result, studies show that it is one that consumes more power [34, 10].

GreenBrowsing aims at extending Chrome, in order to decrease the energy costs of browsing, by taking into account idle tabs (tabs that are open but not being used), as well as taking advantage of the browser API to perform energy-related optimizations. GreenBrowsing will also provide certification of web pages, to ensure that users become aware of which pages are the less green and more resource hungry.

The Document is organized in the following manner: in Section 2 we present the goals that are aimed to be achieved, with this work; in Section 3, the classification and presentation of the relevant related work is done; in Section 4 we present the architecture of GreenBrowsing and in Section 5 we present the metrics that will be used for the evaluation of the system's implementation; in Section 6 we will conclude with some final considerations.

2 Objectives

This work has two main goals. The first one is to *enhance browsing environmental performance*, by developing of an extension of the existing web browser Google Chrome. It is expected to substantially leverage the power consumption rates inherent browsing with Chrome, while still maintaining acceptable levels of availability and performance. This is intended to be achieved through a system that actively monitors tab creation and switching by users, in such way that idle tabs that are less likely to be accessed in a near future should be disposed. The second goal is to provide a means for automatic and *dynamic site and component energy-related certification*, by performing cloud data analytics. This is done by sending important energy and performance data, of each web-site that is accessed, to a system deployed remotely in the cloud. This system will perform the heavy statistical operations needed to certify web pages, releasing the extension running on Chrome from doing such job, not wasting resources for that purpose.

3 Related Work

The next sections will present the work and areas of research that are related to this context of energy-aware web browsing and were more relevant to our proposed work. Section 3.1 will provide a description of how to dynamically manage power consumption. Section 3.2 will present scheduling algorithms to reduce energy losses, in multi-task environments. Section 3.3 will address big data and energy analytic systems.

3.1 Dynamic Power Management

Many of the hardware devices and software components that belong to computational systems are event-driven. Events can be direct commands issued by other components or other kinds of events like I/O interruptions, but in essence they are asynchronous and

lead to intermittent work periods. Between work periods, idle periods take place. However, during idle periods energy is still consumed. In a typical data-center scenario, for instance, when server utilization is below 30%, idle servers still consume the equivalent to 60% of their work-peak power consumption [27]. This asymmetry needs therefore to be dealt with.

Dynamic Power Management, (DPM in short), 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) [32]. These reductions of power dissipation are typically subject to performance and inherent quality of service constraints.

We will start by presenting, in a top-down approach, the characteristics of Dynamic Power Management systems. In Section 3.1.1 we will explain the architecture of dynamic power management, at the highest abstraction level possible. In Section 3.1.2 we will present a classification of DPM systems depending on the different aspects of the DPM architecture elements. Finally, in Section 3.1.3, some DPM solutions will be presented and classified according to the aspects described in Section 3.1.2.

3.1.1 Architectural Overview. Benini et al. [9] 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, taken from [9]). 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.

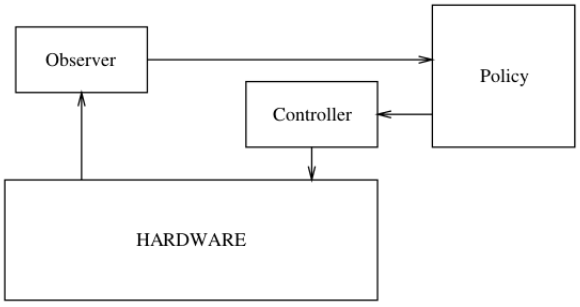

Fig. 1. Dynamic Power Management Architecture.

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 talks directly to devices 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.

3.1.2 Classification of Dynamic Power Management Systems. The functioning of a Dynamic Power Management system is related to certain aspects that depend both on the relations between components and the system under management. The way that

components interact and the way that power management decisions are enforced might penalize performance. In this Section we present a classification that emphasizes the performance penalties of different design choices for dynamic management systems.

3.1.2.1 Power Models. There are some ubiquitous details that influence decision making. These details arise from the way the system under management behaves and reacts to the actions that are performed on it. To that end, policies should be based on proper Power and System models.

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).

3.1.2.1.1 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 behavior 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 modeling simple systems, under near-always-right assumptions, these might suffice in providing good power management capabilities.

3.1.2.1.2 Stochastic Power Models. A stochastic model [23] 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 called 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.

Stochastic models try to solve the problem of Dynamic Power Management in a different way of Heuristic Models. They try to answer the following question: Given the current state of the system, what better (future) sequence of actions could minimize the power consumption, knowing that the system can change amongst different power modes at any given moment? By calculating the probabilities of different sequence of actions and by weighting together with performance costs of applying each action and transiting from one state to another, (among other variables), stochastic models are used to generate policies that execute well under specific performance constraints (but still Heuristic models typically perform better). The problem however is to devise optimal policies, since systems often behave unpredictably.

Amongst the many types of stochastic models, are the widely used Markov Models [21]. In these models the Markov Property [30] 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 $n\text{-}1^{\text{th}}$. This approximation is quite handy, since it just requires the computation of the probability of a certain eventn^{th}, conditioned to the previous $n\text{-}1^{\text{th}}$ one, disregarding all the events observed previously.

Table 1 shows how Markov Models can be classified regarding *observability* of system events and *control* over the system where the Markov Model is applied. In a *partially observable system*, not all states are known beforehand, being discovered dynamically. 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.

	Observable System	Partially Observable System
Autonomous System	Markov Chain	Hidden Markov Model
Controlled System	Markov Decision Process	Partially Observable Markov Decision Process

Table 1. Markov Model classification.

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.

Markov Models can also be further classified according to the cardinality of their time set and state space. Table 2 presents this kind of classification with more examples of stochastic processes.

	Discrete-time	Continuous-time
Discrete-states	Markov Process	Continuous-time Markov Process
Continuous-states	Harris chain	Wiener Process

Table 2. Markov Model Time Set and State Space cardinality classification.

As became apparent in table 3.1.2, more types of stochastic processes exist. We will disregard the continuous-state processes, since the discrete-state ones are more relevant in the context of dynamic power management and the modeled state spaces are oftentimes described as a discrete set.

3.1.2.1.3 Learning Power Models. *"An agent is learning if it improves its performance on future tasks after making observations about the world"* [37]. This proposition is very relevant in 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 example of RL approach is found in the work of Shen et al [39].

A reinforcement learning model consists of three basic elements [39]: a state space that describes the environment, (or system status), an action space that defines the available control knobs and a cost function that evaluates the cost/benefit of different

actions, given the state at which the system is. How these three elements should be defined is determined by the available environment information, the nature of the system under control, as well as the user objectives and constraints. Therefore, it varies from problem to problem.

One common technique of Reinforced Learning is Q-Learning [39] (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. The new state gives the agent a *penalty* which indicates the value of the state transition. The agent keeps a value function $Q_p(s, a)$, (also called *Quality* of the state-action combination), for each state-action pair, which represents the expected long-term penalty if the system starts from state s, taking action a, and thereafter following policy p. Based on this 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?

3.1.2.2 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 behavior, 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, the it is in idle mode;
- if the queue of the Service Provider is not empty, the it is not in idle mode;
- the PM is able to monitor service requests, and conclude the mode of the Service Provider;

Of course, typical real life systems have multiple requesters and multiple providers. This kind of models relax that fact, by considering that all the requests of all requesters come from a single source, and the requests are also enqueued to a single point. Hence, one requester, one provider and one queue.

It is also common to consider more concrete systems as system models. One example is the work of Shen et al. [39] that considers for system model peripheral devices.

3.1.2.3 Adaptation. Power models can be devised statically, before the execution of the policy, never changing 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.

3.1.2.4 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.

3.1.2.5 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. If the system under dynamic power management is an Unix based operating system, for instance, and the components to consider are processes, then a sleep state can be induced through a sleep() system call [4]. Logically, the notion of "reducing the performance of" also varies from system to system. One common way of achieving lower power consumption through performance reductions is through *Dynamic Voltage and Frequency Scaling*. DVFS [49, 17] 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. Such is the case of Intel's SpeedStep [5] and ARM's PowerNow! [1].

3.1.2.6 Policy Optimization. Policy classification can be done with respect to *optimality*. Benini et al. [9] 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*.

The optimality of a given policy is always subject to the system model, in consideration, and the power model itself.

3.1.3 Relevant Dynamic Power Management Solutions. In the work by Qiu et al. [35], the authors describe the problem of DPM as a continuous-time Markov Decision Process, applied to a SRSP system model. Other work, previous to this one, has some disadvantageous characteristics such as (1) considering time as a discrete dimension and (2) no notion of idleness in the modeled system components. This lack of accuracy would contribute to small energy gains because discrete models are limiting when managing real-time applications and because idle states are the ones at which power management actions should be enforced.

To overcome these disadvantages, Qiu et al. devised 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). The way request arrival and request service are modeled with Poisson distribution for the request arrival times, at the Service Requester (SR), and exponentially distributed request service times, at the SP. That lets the PM to be modeled as a event-driven component, thus reducing its decision making overhead, when put in practice.

The overall objective is to put the SP to sleep as soon as it enters the idle state. To do so, a Policy Iteration Algorithm is considered, in order to account for the performance constraints, inherent to the overheads of putting an SP to sleep (when they are idle), and wake the SP up (when they need to serve an SR). 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. [17], the authors prove in a theoretical fashion 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. [19], 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 neighbor 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 [39] 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. As I/O devices they are, their workings are very similar to the Service-Requesters-Service-Providers model (SRSP), described in Section 3.1.2.2. To estimate the quality function $Q_p(s,\ a)$ of each state-action pair, it is considered the expected average power *Power(s, a)* and request delay caused by the action a taken in state s, *q(s, a)*. The expected average power is computed as $Power(s,a) = \frac{(P_{\mathrm{A2B}}T_{\mathrm{A2B}}+P_{\mathrm{B2A}}T_{\mathrm{B2A}})}{2}$, where $\mathrm{P_{A2B}}$ and $\mathrm{P_{B2A}}$ are the power cost of changing from power mode A to power mode B and vice-versa, respectively. The request delay is computed as $q(s,a) = \frac{(q_{\mathrm{A2B}}T_{\mathrm{A2B}}+q_{\mathrm{B2A}}T_{\mathrm{B2A}})}{2}$, where q_{A2B} and q_{B2A} are the average request incoming rate during the power mode switching from power mode A to B (along the execution history of the system) and vice-versa, respectively. In this way, 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. [48] 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 modeled as a SRSP model.

A Semi-Markov Decision Process is similar to Continuous-Time Markov Process, with the exception that the decision maker can choose actions only when system changes state. Therefore, power management actions will be enforced only after the events that change the system's state. Temporal Difference Learning assume 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 chose 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 3 the different algorithms previously presented are summarized according to the classification criteria established in the Section 3.1.2. 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.

Paper	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 3. Dynamic Power Management Schemes Classification.

3.2 Energy-Aware Scheduling

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 [38]. 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.

Before studying different algorithms, two preliminary notions will be given, regarding the nature of multiprocessing systems and nomenclature.

1. From the perspective of scheduling, multiprocessor systems can be classified into (at least) two categories: (1) Heterogeneous processors are different in terms of architectural design. (2) Homogeneous processors are identical in design; hence the rate of execution of all tasks is typically the same on all processors. [53]

2. Each invocation to a task is called a *job*.

Different scheduling algorithms exist to manage a variety of system resources. We will focus on multiprocess systems, since our interest is to study the impact that performance and energetic constraints have on the execution of tasks. We start by studying the characteristics of some classical scheduling algorithms (Section 3.2.1) and, after that, we present energy-aware scheduling algorithms (Section 3.2.2), relevant to the work we pretend to develop.

3.2.1 Classical Scheduling Algorithms. In the *First-Come-First-Served* (FCFS) scheduling algorithm [52], 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 [46] asserts to each job a time-slice where it can run. If a job cannot be completed in a time-slice it will return to the waiting queue and wait for the next time it is scheduled. 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 [20] is an 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.

3.2.2 Reference Energy-Aware Scheduling Algorithms. In the work of Kamga et al. [22], they propose a solution where they extend Xen default Virtual Machine scheduler – *Credit*. The goals are to (1) proportionate power reduction in the execution of several consolidated VMs while (2) respecting the agreed Service Level Agreement (SLA) – maintaining acceptable levels of performance.

Credit has two important parameters: *weight* and *cap*. Weight represents the priority of the VM and cap the CPU usage share, given in percentage. At least one virtual CPU (VCPU) is defined per VM. The scheduler transforms the weight into a credit allocation resource for each VCPU. As a VM runs, it consumes credit. Once the VM runs out of credit, it only runs when other VMs have finished executing. Periodically, it is given more credit to each VM. The role of the extension consists in measuring the total VM CPU load, amongst all VMs, modifying each processor frequency through DFVS, every time the scheduling routine is executed.

The extension of the Credit scheduler is compromised 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. Therefore, the objective of such re-calculations is to avoid performance degradation. 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 [52] where they introduce a job scheduling mechanism that takes the *variation of electricity price* into consideration as a means 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.

Typically, user jobs are submitted to an HPC system through a batch scheduler, and then wait in a queue for the requested amount of system resources to become available. In particular, FCFS with backfilling is a commonly used scheduling policy in HPC, which might waste energy in an arbitrary way.

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.

The scheduling policy is intended to balance energy usage and scheduling performance and it is modeled following a 0-1 Knapsack based policy [45]. Knowing that the overall objective is to reduce the accumulated power consumption during on-peak periods (high system load) and to increase the accumulated power consumption during the off-peak periods (low system load), the policy's goal is to minimize the value of the aggregated power consumption of nodes, during the on-peak period, and to maximize that same value, during the off-peak periods. Therefore the knapsack size is the number

of available nodes, at the time of schedule and the objects that are to be put into the knapsack are jobs.

In the work of Datta et al. [14], 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 multiprocessor system.

The key idea explored in this work is that a task whose context is switched too often may not find valid data in its new core's cache, after being migrated to a new CPU. This task will have a tendency to generate many cache misses. This overhead associated with cache coherence, and with context switching itself, can degrade the performance of a multi-core processor system. The scheduling is done by taking these facts into consideration.

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. For every initialization of a task, a static nice value is given to it, signifying its priority. Typically, tasks that are known to be CPU intensive require a lower nice – more priority. 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 the nice value.

3.3 Energy-related Certification and Analytics on the Cloud

The academic efforts to provide a means to certify web-pages in terms of its resource consumption are nearly inexistent, as far as we were able to find, even though it is a plausible idea to explore.

Here we start by analyzing the current solutions that attribute some sort of energetic rating to computational systems (Section 3.3.1). We then move to the cloud and big data systems domain (Sections 3.3.2 and 3.3.3) 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.

3.3.1 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 [40] to certify mobile devices, regarding their energetic performance. The idea of a green mobile certification is to use a set of test cases, which represent scripts of different mobile use patterns, to evaluate a mobile device. 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 [7] – 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. For that effect, users are prompted to specify which classes of

software they want to test. 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 [12] 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.

3.3.2 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. identify 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) [44].

- 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 [44].

If we also consider that the system is compromised 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 DataStore and application Runtime are placed at the client end of the system. There is no remote end.
- *Local-DataStore-Remote-Runtime* (LDRR) - The DataStore is placed at the client side while the application Runtime is executed remotely.
- *Remote-DataStore-Remote-Runtime* (RDRR) - Both the DataStore 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 [44]. On the other hand, consumer-centric approaches emphasize consumer preferences regarding the way their data are handled [47], by integrating their preferences in the decision-making of the services provided.

3.3.3 Relevant Energy-related Big Data Analytics Systems. In the work of Lachut et al. [24], 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. Therefore only a small portion of the devices is installed at consumer houses. These devices will provide only the necessary metrics in order to statistically determine the energy consumption of each appliance, using a technique called Additive Factorial Hidden Markov Model.

The system is compromised by (i) Home Components that gather energy counters and send them to a (ii) BackEnd Server, which is responsible to apply the statisti-

cal methods necessary to measure the energy wasted, at different levels of granularity. Consumers can also visualize the energy wasted in their mobile devices through a Smart-phone Application, since the BackEnd Server provides a RESTfull API that these applications use.

In the work of Lee et al. [25], 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. [44], 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 analyzed 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. The system is separated in three main components: (i) Gateway, (ii) VHome and (iii) a variety of Applications. The (ii) Gateway is an home-resident and consumer-controlled component that collects home energy production and usage data that are uploaded, over a secure connection, to the cloud-based virtual home: the VHome component. It also provides an interface to allow the house owner to control devices in his/her house from Internet-connected devices. VHome is a virtualized execution environment hosted on a IaaS cloud provider. This (ii) VHome component is comprised by the Data Store and Application Runtime components described in Section 3.3.2. The (iii) Applications that can access the Data Store freely are the ones that belong to the VHome application runtime. In order to allow other applications to access the Data Store, such as third party applications that can provide different analysis algorithms, privacy protection mechanisms (PPMs) are enforced. This PPMs pre-process data before it is transferred out of the VHome, by employing mechanisms like noise addition to the data transfered out of the VHome to these applications.

In the work of Balaji et al. [8], the authors present a system called ZonePAC, for the energy measurement of modern houses with Variable Air Volume (VAV) type heating, ventilation, and air conditioning (HVAC) system and energy consumption feedback provision to the house occupants through a web application. The system basically 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 [50]. The communication is possible because a BACnet Connector (a user Computer) is adapted to the BACnet network. This network is formed of sensors and the BACnet Connector that communicate over a BACnet protocol. The BACnet Connector makes use of the RESTful BuildingDepot API to communicate energy counters gathered. BuildingDepot will act as a energy counter dissemination broker, since it informs the web applications subscribed on the available web services of new energy measurements. Finally, the users might check what are the energy consumption indexes of their in-house HVAC systems.

In the work of Oliner et al. [31], the authors propose Carat, a system for diagnosing energetic anomalies on mobile devices. This system consist in a client application, running on a client device, to send intermittent, coarse-grained measurements to a server, which correlates higher expected 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. There, the samples from client devices are sampled and analyzed, aggregating the consumption of various mobile devices at the moment of analysis.

Table 4 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 5 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.

System / **Feature**	Balaji et al.	Lachut et al.	Lee et al.	Oliner et al.	Singh et al
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 4. Big Data System Features.

System	Energy Data to Visualize	Design Rationale	Business Rationale
Balaji et al.	Home Energy Consumption	LDLR	Utility-Centric
Lachut et al.	Home Energy Consumption	RDRR	Utility-Centric
Lee et al.	Home Energy Consumption	RDRR	Utility-Centric
Oliner et al.	Mobile Device Energy Anomalies	RDRR	Utility-Centric
Singh et al.	Home Energy Consumption	RDRR	Consumer-Centric

Table 5. Big Data System Classification.

3.4 Analysis and Discussion

In this related work section, different energy and software related topics were covered, in order to understand how could a browser power management solution be devised. For each of the related work topics presented, some final considerations will be provided. These considerations will help to devise a suitable architecture for GreenBrowsing.

In Section 3.1 the trade-offs of Dynamic Power Management are explored, 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 – the exploitation of idle periods to optimize power consumption – resembles some of the intent of GreenBrowsing: to reduce the power wasted by idle tabs. With this in mind, it is important to consider that the more the policy adapts, the better the decision making is. However, adaptable solutions incur in bigger overheads and sometimes offer similar energy gains compared

to more static solutions. Moreover, it is also important to have power management components that communicate asynchronously between themselves to avoid performance degradation.

In Section 3.2, scheduling was presented taking 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. It is important to separate concerns in a balanced fashion, in order to assign similar workloads to different tasks, in terms of the computational and resource access intensiveness of each job.

In Section 3.3, emphasis was given to the fact that it is desirable to move expensive and resource intensive computations to a remote system (e.g. in the cloud), when it comes to energy management. The data needed to do those computations can sometimes disclosure private details of users. In this way, the information sent for remote processing should be as few, and as protected, as possible. In order to reduce the waiting time of resource processing on client applications, running on client devices, the server/cloud counterparts should execute as fast as possible to reduce the turnaround times of sending energy-related data to remote systems, processing it there and receiving it back. Therefore, the performance of remote data processing matters. Furthermore, remote systems should scale with the processing requests they receive. So scalability is also important to take into account.

4 Architecture

There are two major sub-systems that comprise the GreenBrowsing architecture: a Google Chrome Extension that will act as a power manager and will manage resource consumption of tabs (Section 4.1), and a Web Page Certification BackEnd, to be deployed as a prototypical big data analytics system on the cloud (Section 4.2).

The Architecture of GreenBrowsing will be presented following the *Modules* and *Component and Connector* View Styles [16], loosely.

4.1 Chrome Extension Sub-System

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

Since performance is an issue that determines user experience, the GreenBrowsing Chrome Extension architecture is designed to circumvent performance issues, exploring the parallelism at different functionality levels. Furthermore, in order to properly separate concerns, a modular architecture will be adopted. This will allow the extension itself to be more easily extensible and easily portable to other modern browsers, such as Mozilla Firefox.

4.1.1 Modules and Run-Time Components Description A Layered View of the Chrome Extension is presented in Figure 2. Each layer *uses*, exclusively, the layer(s) beneath it. Each of the modules is described as follows:

– **Observer-Controller-Adapter (OCA) -** It provides interfaces for gathering performance counters of each running tab and the process(es) it is associated with.

It will also provide interfaces for issuing commands to tabs and the operating system itself. Moreover, it will be possible to adapt web pages through this module. Examples are the removal of images or sound.

- **Certification FrontEnd -** Here is the code of the network communications that will need to be carried out with the Certification BackEnd. Through it, performance data of web pages will be sent to the cloud.
- **Certification Renderer -** This module embodies the functionality needed to render energy-related rating of each web page, based on its certification, serving presentation purposes mostly.
- **Policy Enforcer -** Here the power reduction algorithm will be implemented. This module will need to use the OCA interface, to gather performance counters and to issue content adaptation and power reduction related commands.
- **Web Page Certifier -** This module will have code to fetch performance counters, through the Observer-Controller-Adapter. It will also interface with the Certification FrontEnd to send the counters gathered to the BackEnd (for energy-related certification of web pages). Communications with the Certification Renderer are done to inform the user of each web page certification.
- **Profile Manager -** This module will have the code for the graphical interface the user might use to further tune GreenBrowsing to his/her preferences. Interfacing with the Policy Enforcer is done to communicate user preferences.

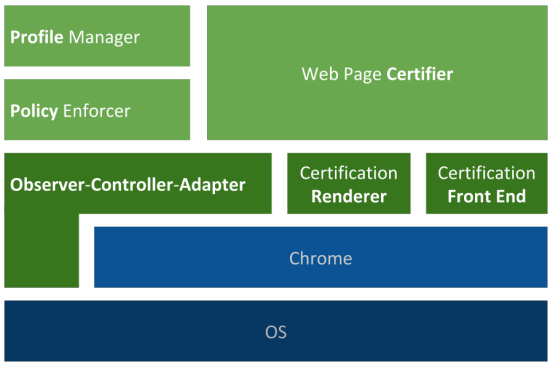

Fig. 2. Layered View of The Chrome Extension.

In terms of components, the code of the Policy Enforcer will be run by a dedicated set of threads. The execution of the OCA control and content adaptation code will be done by a different set of threads, in order to speed up the execution of the extension. This allows the policy enforcement to be executed in parallel with the control and content adaptation of tabs/pages (which otherwise could cause significant delays in the policy threads execution).

There will be also a dedicated set of threads running part of the Web Page Certifier code and part of the FrontEnd code that will deal with the issuing of energy related data to the certification BackEnd. The other part of the code is run by other set of threads that will react (asynchronously) to the incoming certification rankings that come from the BackEnd certification system, avoiding waiting busily for those responses. Once

received the certification stamps, these threads will also be responsible for running the Certification Renderer code, for the visualization of web page energy-related certification.

In order to avoid unnecessary round-trips, a cache of certification stamps will be maintained in the chrome extension.

4.1.2 The Tab management Policy. We consider that there is no best way to approach the problem of managing tabs for achieving power gains.

At a first glance, the complexity of the problem seems to require stochastic or machine learning-based techniques to suitably approach it. However by doing so, the utility and accuracy of these techniques could be rapidly questioned. Specially due to the relative *unpredictability of user actions* and *variability of browsing habits*. But even if those techniques were accurate, they are nonetheless computationally intensive in the great majority cases (as discussed in Section3.1), introducing higher power consumption rates themselves.

Having this into account, we conclude that we should approach this power management problem through simplistic heuristics, that offer a smaller implementation overhead compared to stochastic or machine learning techniques. Specially because one of the requirements of this system is to interfere the least possible with user experience. Therefore, we make only two assumptions regarding general browsing behavior that will be considered in the policy to use:

- **Last Time Usage.** Tabs that were accessed more recently are more likely to be accessed again and therefore will have a lower probability of being discarded. In this way, tab management will make use of a Least Recently Used list for tab disposal.
- **Distance to other Tabs.** 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 less probability of being discarded.

Since we want to manage the power consumption of tabs, we will infer power consumption from resource consumption. In particular, the memory each tab is using, the CPU intensiveness (in terms of load and number of cycles), and disk accesses in terms of the read/write accesses, all of which on average and over fixed periods of time.

Moreover, we intend to leverage recent work in our research group, addressing resource management and elastic scheduling in cloud infrastructures, that take into account relative efficiency of scheduled resource usage, with application progress monitoring, factoring in perceived utility depreciation by users when resources are sub-allocated [43, 42, 41]. We will address similar constrains in this work, as the best possible performance is only achievable with high resource usage, and we want users to obtain experiences that combine adequate performance within the intended energy consumption profile.

4.2 Analytics & Certification BackEnd Sub-System

The Certification BackEnd Sub-System has the objective of *providing a clear and meaningful notion of how much energy web pages consume*. It also will certify domains, as a way to alert users of web-sites that generate resource hungry, power consuming web pages.

As a way to reduce the computational intensiveness required by the extension, the energetic certification of web pages is moved to a separate sub-system, that is intended to be deployed in the cloud.

4.2.1 Components of the Certification Sub-System. The Certification BackEnd will have, at least, the following components (as depicted in Figure 3):

- A *Network Communication task* that receives energy-related web page certification requests and forwards these requests to specialized workers (to avoid service bottlenecks and enhancing the scalability of the system regarding the treatment of requests);
- *Analytics Certifier tasks* that do the work of certifying a given page, according to a specific certification model.
- A *Certification Modeler task* that adjusts the certification model, having into account all the resource data sent from the extension subsystem. For performance purposes, this design sub-intends 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 to the Modeler Task, as soon as they are ready.
- An *Analytics Data Store* that stores the models used in the certification of pages and tuples with information relative to the performance counters of each page;

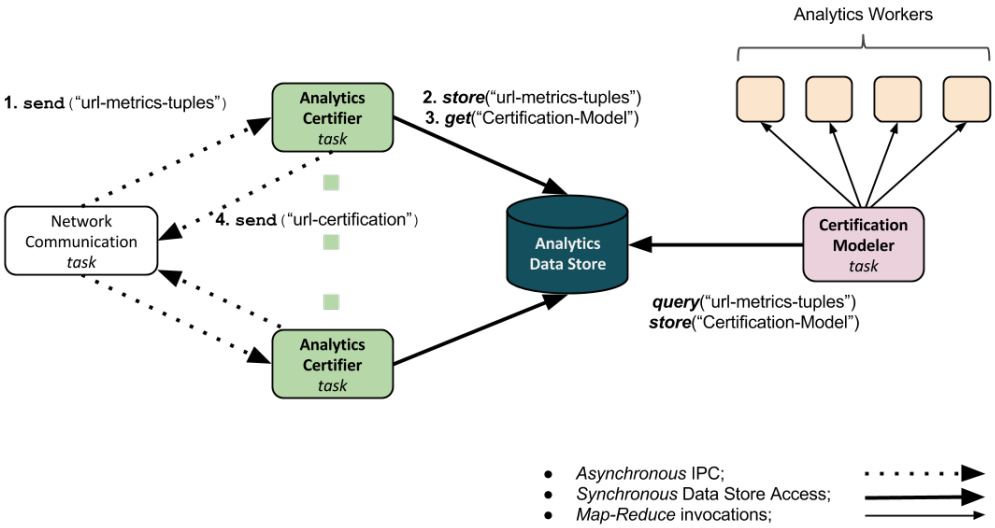

Fig. 3. The Communicating tasks of the Backend.

4.2.2 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 Chrome Extension. For each page, the metrics considered will be (1) CPU clock cycles, (2) Main Memory usage and (3) the read/write latency of disk accesses, to process each page. These metrics were chosen because they were proved to be highly related to the power consumption, in different settings ([36], [11], [33], respectively).

The certification will be done at the level of the *web page* and *domain*. Therefore, the information sent from the chrome extension to the Certification BackEnd will be a 5-tuple <*domain*, *page-url*, *number-of-cycles*, *memory-usage*, *r-w*>.

These metrics will be sent to the BackEnd, after a page is rendered (accounting for the resource consumption of JavaScript just-in-time compilation and HTML+CSS

processing) and before the tab of a certain page is disposed by the power management threads (accounting for the resource usage due to user activity).

4.2.3 The Certification Model. The problem of asserting a certification rank to a page, given its performance counters, is an *Unsupervised Learning* problem since, in essence, we want to devise a function with known inputs and unknown outputs [37].

Apart from devising such function, two questions need also to be answered: (i) How to map the performance counters to certification ranks? (ii) Given two ranks, which one dominates, or better yet, which one represents the *greener* rank?

To solve the first problem we employ a clustering algorithm known as the *Expectation-Maximization* (EM) algorithm [15] in order to find no less than 8 categories of certification (clusters), in a 3-dimensional space that comprehends one dimension for the CPU clock cycles, one for the Memory usage and another for read-write operation latency. The (EM) algorithm is an efficient iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. In ML estimation, we wish to estimate the model parameter(s) for which the observed data are the most likely.

Each iteration of the EM algorithm consists of two steps: The E-step, and the M-step. In the expectation, or E-step, the missing data are estimated given the observed data and current estimate of the model parameters. In the M-step, the likelihood function is maximized under the assumption that the missing data are known. The estimate of the missing data from the E-step are used instead of the actual missing data.

Convergence to the Maximum Likelihood function is assured since the algorithm is guaranteed to increase the current function at each iteration. In this way, the algorithm steps are repeated, iteratively, until convergence (which is obtain by approximation, when no significant relative gain is obtained).

At the end of the EM algorithm, 8 distinct clusters are obtained. Each one of these clusters is associated with a vector called centroid. We assume that given two categories whose cluster C_i and C_j have centroids c_i and c_j, respectively, if $|c_i| > |c_j|$ (where $|c_i|$ denotes the dot period of c_i) then the vectors that are members of the cluster C_i are more power consuming than the ones of C_j. In this way we are able to classify, and therefore certify, pages regarding energy consumption.

To certify domains, the arithmetic mean and the geometric mean of the page classifications (obtained with the method explained previously) are computed.

4.2.4 Privacy Concerns. In order to preserve user privacy, the URLs and domain identifiers sent to the BackEnd will be partially anonimyzed by hashing them independently. This will also enable fast indexing/search, after being stored remotely, while providing significant privacy, since it requires extensive brute force to extract the URL, given an hash of it.

5 Methodology and Evaluation

The evaluation of GreenBrowsing will be done with respect to two evaluation vectors: (1) the energy gains by the usage of the extension and (2) perceived latency by the user.

Since it would be difficult to differentiate the actual energy wasted by the browser with idle pages, with and without our extension, from noisy energetic patterns caused by other applications and system activity, other metrics than Joules or Watts per second will be taken into consideration for the evaluation of energy gains. Therefore, energy gains

will be measured indirectly through other more accessible and easy to fetch metrics, such as CPU load and memory usage.

Evaluation Metrics. For the the energy gains by the usage of the extension, the metrics to consider will be: (i) *CPU load per tab*, (ii) *memory usage per tab* and *I/O reads and writes per tab*.

For measuring the perceived latency by the user, the (i) *overall time to load pages* (i.e. fully render pages) will be considered. It will also be considered the (ii) *processing time at the BackEnd*, in order to account for late certification stamps.

Set Up and Testing Method. In order to conduct the evaluation of the resource usage optimization that GreenBrowsing provides a set of typical web pages will be used. These compromise web pages of news, social network, sports, mail clients and multimedia in order to provide a rich and varied Web page suite to test.

To test how the extension behaves while navigating through different tabs, different Browsing Behavior Policies will be used. In particular, the following navigation policies will be employed:

- (A) *round-robin selection* to navigate sequentially from tab to tab;
- (B) *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 bar to the last;
- (C) *random tab selection* to assess if there are actual gains compared to the directed policies A or B.

In order to observe the effects of this methodology and extract the measurements intended for each metric, the Performance Profiling Capabilities with the Timeline monitor [3], from the Chrome DevTools suite, will be used.

To measure the latency induced by the extension, the instrumentation of web pages will be done at critical points, in order to measure the time period that goes from the moment a certain idle tab is reselected to the moment it is completely rendered.

6 Conclusions

In this report, we started by introducing the relevant related work to the system we propose. We classified Dynamic Power Management systems, Scheduling algorithms and Big Data systems for energy-related processing and identified their limitations. We also identified the design choices and characteristics that will allow us to devise an effective power management and energy-related web page certification system for Chrome users, having into account each of the cases studied. In that way, we presented the architecture for GreenBrowsing. Finally we presented the metrics and method that will be used to evaluate our solution, regarding the effective energy and resource usage spared by the use of our system and its performance.

References

1. AMD PowerNow!™ Technology. http://www.amd.com/us/products/technologies/amd-powernow-technology/Pages/amd-powernow-technology.aspx, accessed: 2013-11-29
2. Chrome browser. https://www.google.com/intl/en/chrome/browser/
3. Chrome devtools. https://developers.google.com/chrome-developer-tools/
4. sleep(3) - Linux man page (2001), http://linux.die.net/man/3/sleep
5. Wireless intel speedstep power manager (2004)

6. Amsel, N., Ibrahim, Z., Malik, A., Tomlinson, B.: Toward sustainable software engineering (nier track). In: Proceedings of the 33rd International Conference on Software Engineering. pp. 976–979. ICSE '11, ACM, New York, NY, USA (2011)
7. 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)
8. 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)
9. 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)
10. Bianzino, A.P., Raju, A.K., Rossi, D.: Greening the internet: Measuring web power consumption. IT Professional 13 (2011)
11. Bircher, W.L., John, L.K.: Complete system power estimation using processor performance events. IEEE Transactions on Computers 61(4), 563–577 (2012)
12. Camps, F.: Web browser energy consumption (2010)
13. 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)
14. 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)
15. 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)
16. Garlan, D., Bachmann, F., Ivers, J., Stafford, J., Bass, L., Clements, P., Merson, P.: Documenting Software Architectures: Views and Beyond. Addison-Wesley Professional, 2nd edn. (2010)
17. Gerards, M., Kuper, J.: Optimal dpm and dvfs for frame-based real-time systems. TACO 9(4) (2013)
18. 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)
19. 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. pp. 288–295. DSD '12, IEEE Computer Society, Washington, DC, USA (2012)
20. Jansen, P.G., Mullender, S.J., Havinga, P.J., Scholten, H.: Lightweight edf scheduling with deadline inheritance. Tech. rep., University of Twente. May (2003)
21. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2) (1998)
22. Kamga, C.M., Tran, G.S., Broto, L.: Extended scheduler for efficient frequency scaling in virtualized systems. SIGOPS Oper. Syst. Rev. 46(2) (2012)
23. Klebaner, F.C.: Introduction to stochastic calculus with application (3rd edition) (2012)
24. 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)
25. 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)
26. 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)

27. Meisner, D., Gold, B.T., Wenisch, T.F.: Powernap: Eliminating server idle power. SIGARCH Comput. Archit. News 37 (2009)
28. Miettinen, A.P., Nurminen, J.K.: Analysis of the energy consumption of javascript based mobile web applications. In: MOBILIGHT (2010)
29. Murugesan, S.: Harnessing green it: Principles and practices. IT Professional 10 (2008)
30. Norris, J.R.: Markov chains. Cambridge Series in Statistical and Probabilistic Mathematics (1998)
31. 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)
32. 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)
33. 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. pp. 24–35. SEUS '09, Springer-Verlag, Berlin, Heidelberg (2009)
34. Patel, S., Perkinson, J.: Fraunhofer report - the impact of internet browsers on computer energy consumption (2013)
35. 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)
36. 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)
37. Russel, S., Norvig, P.: Artificial intelligence: A modern approach (3rd edition) (2009)
38. 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)
39. 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)
40. 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)
41. de Oliveira e Silva, J.N., Veiga, L., Ferreira, P.: A2ha - automatic and adaptive host allocation in utility computing for bag-of-tasks. Journal of Internet Services and Applications (JISA) 2(2), 171–185 (September 2011)
42. Simão, J., Veiga, L.: Qoe-jvm: An adaptive and resource-aware java runtime for cloud computing. In: 2nd International Symposium on Secure Virtual Infrastructures (DOA-SVI 2012), OTM Conferences 2012. Springer, LNCS (September 2012)
43. Simão, J., Veiga, L.: Flexible slas in the cloud with partial utility-driven scheduling. In: IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013) - Best-Paper Award Runner-up. IEEE (December 2013)
44. 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)
45. T. H. Cormen, C. Stein, R.L.R., Leiserson, C.E.: Introduction to Algorithms. Higher Education, McGraw-Hill (2001)
46. Tanenbaum, A.S.: Modern Operating Systems. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edn. (2007)
47. W. Liu, K.L., Pearson, D.: Consumer-centric smart grid. Innovative Smart Grid Technologies pp. 1–6 (2011)
48. 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)
49. Weiser, M., Welch, B., Demers, A., Shenker, S.: Scheduling for Reduced CPU Energy (1994)

50. 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)
51. Yan, H., Lowenthal, D.K., Li, K.: Ace: An active, client-directed method for reducing energy during web browsing. In: Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video. NOSSDAV '05, ACM, New York, NY, USA (2005)
52. 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)
53. Zhuravlev, S., Saez, J.C., Blagodurov, S., Fedorova, A., Prieto, M.: Survey of scheduling techniques for addressing shared resources in multicore processors. ACM Comput. Surv. 45 (2012)

A
Work Planning Table

Achievement/Deliverable	**Month – Expected Time to Deliver**
Explore Chrome Extensions Technology (API, frameworks)	February – 1 week
Chrome Extension Base System - Observer-Controller-Adapter (implemented & tested) - Profile Manager (implemented & tested) - Certification Renderer (implemented & tested) - Certification FrontEnd (mocked up) - Policy Enforcer (prototyped)	February/March – 2-3 Weeks
Chrome Extension Fully Implemented - Certification FrontEnd (implemented & tested) - Policy Enforcer (implemented & tested) - Integration Testing of Chrome Extension Components - Chrome Extension Documentation	March/April – 2-3 Weeks
Chrome Extension Evaluation (as described in Section 5)	April – 2 Weeks
Explore Statistical Frameworks, **Big Data Frameworks**	April/May – 1 week
Certification BackEnd Base System - Code for the Network Certification Task (implemented & tested) - Code for the Analytics Certifier Tasks (mocked up) - Code for the Data Store Management (mocked up)	May – 1-2 Weeks
Certification BackEnd Fully Implemented - Code for the Analytics Certifier Tasks (implemented & tested) - Code for the Data Store Management (implemented & tested) - Code for the Certification Modeler Task (implemented & tested) - Integration Testing of Certification BackEnd Components	May/June – 3 Weeks
Evaluation and Testing of the Entire System	June – 2 Weeks
Dissertation Writing	June/July – 4 Weeks