Teaching The Microprocessors Systems Focused on Societal Challenges: Designing of Performant Cache Replacement Algorithms as Green Information Technology (IT) Solution

Adrian Florea[1]
"Lucian Blaga" University of Sibiu, Computer Science and Electrical Engineering Department
Sibiu, Romania
adrian.florea@ulbsibiu.ro

Journal of Digital Information Management

ABSTRACT: *In the last decades, beside Education, Information & Communication Technology was, and will continue to be, the engine of evolution and development of society. The main goal is to achieve a better world for all of us, raising living standards using ubiquitous computing systems of any type (desktop, laptop, server, high performance computing data centers, smart phones and other portable multimedia devices, embedded systems, etc.). In order to keep this evolutionary trend should be understood the technical challenges targeting each component of the computer system, in particular the microprocessor – as known as the "computer brain". The formal learning of discipline entitled "Microprocessors Systems" supposes that for addressing the technical challenges of microprocessors it should be discussed, analyzed and understood the current major societal challenges, and mainly the requirement of cutting down the energy consumption and of reducing our carbon footprint. In this sense it aims to develop applications and computing systems that fall within the Green ICT paradigm that means using hardware/software technology efficiently in the same time with providing economic viability, social responsibility and clean environmental impact. The work that underlies this research paper fall into the category of "Green by IT" aiming to reduce the environmental impact of operations using IT and by efficient resource usage management. More precisely, we intend to present a solution of reducing the video traffic on the Internet by designing an appropriate algorithms of cache blocks replacement which reduces the cache miss accesses in social media data storage centres. This not only decreases the video-retrieval time pleasing the user, but also will reduce the energy consumption with world-wide social benefits. By implementing performant cache replacement algorithms we will accelerate the search process of videos in YouTube and we will measure the Internet traffic exemption (in MB) if the cache miss rate decreases (for each particular algorithm) and we will also estimate how much energy is saved by the data centers of search engines. This has a very positive social impact by contributing to maintaining a clean environment, given that the resource requirements (storage and network bandwidth) of social media are growing.*

Subject Categories and Descriptors
[C.5.3 Microcomputers]: Microprocessors; **[K.3.2 Computer and Information Science Education]; [H.3.5 Online Information Services Web-based services]**

General Terms
Microprocessors, Web-based Education, Internet, Information and Commmunication Technology

Keywords: Cache, YouTube, Microprocessors System, Societal challenges, Teaching, Energy, Environment.

Received: 11 December 2016, Revised 14 January 2017; Accepted 24 January 2017

1. Introduction

The scientific domains like Microprocessors Architecture or Microprocessors Systems are extremely complex. Some researchers considered the microprocessor as the most complicated device that has ever been designed, manufactured by tens of engineers working together for several years, and compared it with a metropolis implemented on 2 square centimeters. The continuous expansion of microarchitectures which nowadays embed more than 2 billion transistors on chip has led to a hard to control and understand complexity. Dealing with this is possible only by abstraction. For this reason, it is more difficult to explain certain theoretical concepts like caches, branch prediction, out-of-order and speculative execution, power consumption, and the interactions among the architecture components without visual aids. Most of the simulators are dedicated to solve and visually express only a specific architectural task and cannot express a holistic view and the interaction between all components. For example, due to this complexity, although the designers and researchers efforts are to emphasize the processor kernel activities, many times they ignore the branch prediction and cache memory simulation. Recent teaching strategies are based on sophisticated visual software simulators used to explore the impact of different processors configurations on performance, energy or thermal dissipation or to translate all complex processing mechanisms in relevant and easy to understand information for students. These approaches represent a formative necessity since the topic of microprocessors architecture is mainly tackled by teachers in a descriptive manner. Even the author of this work developed such visual simulators during its research activities [1-3].

However, as far as I know, none of the existing simulators do not analyse the impact of concepts / algorithms from Microprocessors Architecture domain, on the environment. For better fixing of knowledge in any scientific field, we believe that teaching theoretical concepts must be correlated with their practical applicability. Thus, the main goal of this paper is to exemplify how a concept specific to Microprocessors Architecture, namely, the replacement algorithms of conflicting blocks from cache memories can be taught through explaining the impact they have on the dissipated energy and on carbon dioxide (CO2) footprint when are used in servers of data centres which support social media networks. Carbon footprint reflects the environmental impact of a product or service over its entire lifetime [4]. For example, HIPEAC report shows that the CO2 footprint of a desktop computing running for an hour is 60g and a single Google query produces 7g of CO2 [5]. Other studies exhibit CO2 emissions as a global warming potential over 100 years [4, 6].

According to many research studies [4, 5], Europe and the entire World face severe societal challenges especially in energy consumption, environment and security. Global warming with its disastrous effects represents one of the biggest and serious problem of this century. It concerns large and important organisations such as the World Wide Fund for Nature and the United Nations that emphasize the importance of globally reducing energy consumption and of carbon footprint, and of protecting the planet for future generations. Moreover, the European Union (EU) through the Research and Innovation program Horizon 2020, has allocated 80 billion Euros funds for themes like cleaning environment and energy efficiency as societal challenge [7].

Despite the key role in achieving these objectives of Information and Communication Technologies (ICT), ironically, this has a negative impact on the environment, too. ICT is responsible for the amount of energy consumed by hardware engineering equipment, personal computers and data centres, due to the increasing number of devices as well as to network expansion, and software processes and services, etc. In 2010, the computing systems and specially the servers of web browsers consumed the same amount of energy as civil aviation, which was around 1.5% ÷ 2% of the global energy consumption [5]. Also, Ericson researchers have shown that ICT contributes around 2% of the global CO2 emissions and for approximately 8% of the EU's electricity use [4].The Greenpeace specialists [8] warns that the energy consumption of data centres globally will grow from 330 billion kWh in 2007 to 1012 billion kWh by 2020, causing associated CO2 equivalent emissions of 533 megatons which negatively affects the climate.

Beside hardware development, the software expansion occurred. New applications require more hardware resources. For example, performing the same simple task in Office 2010 under Windows 7 requires 70 times more internal memory and 15 times processing power than running in Office 2000 under Windows 98 [9].Also, the software development has created new computer programs used to assist the users in many forms of communication like chat, email, webinars, http requests, media resources and mobile communications available through the Internet. Thus, the ICT specialists forecast that in 2020 video streaming and displaying will represent around 55% of all global mobile data traffic, requiring huge capacity demands on future's networks [6]. This represent an additional reason for HIPEAC specialists to recommend reducing Internet traffic by making local computations instead of transferring data because communication consists in the main source of waste of energy [10].

The solutions to reduce energy consumption follow two directions [4, 5]. On the one hand, research is needed to find alternative secure and sustainable energy sources, for replacing the existing ones.Wind power obtained from smart interconnected turbines systems, placed on earth or on floating platform, is regarded as one of the largest source of sustainable energy [11]. It replaces electricity produced by burning large amounts of coal with benefits on energy savings and by maintaining a cleaner air. On the other hand, it should be significantly reduced total energy consumption of electronic devices. Energy

efficiency practices must be applied in the following areas:

• *Green ICT* solutions such as virtual meetings, improvement in logistics, dematerialization of activities, intelligent transport systems, smart grids, sustainable management and e-government politics of smart cities.

• Efficient algorithms and programing techniques for reducing energy consumption of commercial web search engines and energy conservation in wireless sensor networks.

• Environment conditions for keeping it clean: monitoring parameters (speed, barometric pressure, temperature, polution), optimizing the efficiency of cars engines, reducing or optimizing the traffic.

• Ventilation, cooling and electrical systems applied to data centres.

The main contribution of this work shows that, after learning of concepts specific to microprocessors, students will become more sensitive to vital social problems like energy consumption and carbon footprint of software applications. We present a solution of reducing the video traffic on the Internet by designing an appropriate algorithms of cache blocks replacement which reduces the cache miss accesses in social media data storage centres. This not only decreases the video-retrieval time pleasing the user, but also will reduce the energy consumption with world-wide social benefits. By implementing performant cache replacement algorithms we will accelerate the search process of videos in YouTube and we will measure the Internet traffic exemption if the cache miss rate decreases and we will also estimate how much energy is saved by the data centres of search engines and also what amount of CO2 emissions is avoided. This has a very positive social impact by contributing to maintaining a clean environment, given that the storage and network bandwidth demanding of Social Media are growing. Thus, educational impact of our work overcomes the problem of teaching itself of technical and theoretical concepts, highlighting the social dimension of eLearning.

The organization of the rest of this paper is as follows. In section 2 is described the essential background on *Green IT* and Cache memories then briefly is reviewed some state of the art papers related to this study. Section 3 illustrates the simulator software architecture, the cache replacement algorithms, the simulator user interface and the software developer's vision. In section 4, we present some simulation results. Finally, section 5 suggests directions for future work and concludes the paper.

2. Background And Related Work

In this section, we briefly describe the essential background on *Green IT* and Cache memories then present some state of the art papers related to this study.

2.1 *Green IT*: solution in reducing energy

The importance and sustainability of software engineering including environmental aspects (*Green IT*) has been lately highlighted in many scientific papers. A comprehensive work about hardware but especially software sustainability and many definitions for (software) sustainability and *Green IT* was done in [7]. Sustainability supposes the development for satisfying the present needs, but without risking that future generations would not be able to satisfy their own needs. The software developers of systems, products, industrial or Web applications, data centres should be aware about their responsibility of creating, combining or using energy efficient and more environmentally friendly software products. Many applications require high energy consumption due to poor design, inefficient algorithms, and clumsy written code. In order to improve the software energy efficiency, it must be applied best practices starting with design and development stages.

In [12] the authors shows the effect of code cohesion techniques on code complexity and efficiency, number of instructions executed, memory size requirements, and energy consumption. There are considered four classes of code optimizations: loops, functions, operators, control structure in order to reduce the miss rates in Instruction and Data Cache memories. Somewhat targeting the same objective, in our work we implement advanced cache replacement algorithms, to reduce the cache miss rate and then, by applying some data analytics considerations proved in previous case studies [13], we determine the reduced energy and CO2 emissions.

Another approach for reducing energy consumption by mobile applications was proposed in [14]. The author's solution does not involve changing their source code but, by analyzing the program and applying transformations, to offload parts of its functionality to the cloud, using an adaptive system that determines an optimal offloading strategy at runtime. Having a similar goal, in this paper we track to reduce (video) data traffic through the Internet and measure the energy reduction using the cache concept and the use of replacement algorithms for cache conflicting blocks, based on finding popular videos and applying Pareto-based LFU/LRU algorithms.

In [13] the authors make a classification of programming techniques and common practices applied on different types of computing systems, aiming energy impact evaluation of software applications. They present results of empirical validation of two best practices: put application to sleep and use efficient queries applied on open source software. From their work we borrowed the information that, according with Google, the average amount of energy used by Google servers to transfer 1 MB of data across the Internet is roughly 0.01 kWh. Applying this, for the popular YouTube video "Gangnam Style", which had more than 1.7 billion visualizations, is obtained the yearly energy demand of a city of 22000 inhabitants, quite worrying. In present work, we show that implementing performant cache replacement algorithms decreases the video data

traffic across the Internet, and significantly reduces the energy with up to 5.33 GWh in 20 weeks.

2.2 Cache replacement algorithms

According to definition given by American Webster dictionary [15], cache is "*a safe place for hiding or storing things*". Cache memories represent ubiquitous mechanisms found in nowadays microprocessors, dedicated to reduce the technological gap between processor speed and the memory system latency. Due to their importance, caches are fundamentals in specific curricula of microprocessors architecture. The "cache" concept was first introduced in 1965 by Maurice Wilkes, professor at Cambridge University, a computer science pioneer, winner of several awards including Turing Award (1967) and Eckert–Mauchly Award (1980) [16].The first commercial system that used cache memories was IBM 360/85 (1968) [17].Cache memory efficiency is based on two statistical principles, which intrinsically describe the concept of computer program: the spatial and temporal locality. The temporal locality specifies that is a high probability that an instruction, which is now processed by the CPU, to be processed again in the immediate future. The space locality states that there is a high probability that data located in the immediate vicinity of the data currently accessed by the CPU, to be accessed in the near future.

The concept of cache proved to be fertile not only in hardware but also in software having large applicability. For example, in operating systems, the Virtual Memory mechanism use Translation Look-aside Buffer that behaves as a cache for pages table. In computer networks, the server's operating system keeps, as soon as it finds, in a temporary database of Domain Name System, seen as local cache, mappings of symbolic-numeric addresses regarding to all recently visited websites. In the database management system, MySQL will keep in cache the result of a query and, when will receive the same query, will return the result from cache without querying the tables again. The search engines and Internet browsers store images in a local cache such that, at each refresh event, a page or an image will be taken from cache and will be not accessed the server again. On temporal and spatial locality principles is based even the people behavior when they access YouTube videos: if somebody likes a certain video then he will play again and also the browser will suggest some other videos situated in the spatial vicinity of it (same webpage). In the YouTube case, the cache is represented by MemCached [18] which is a memory caching system intended to speed up dynamic web applications and is in RAM located. MemCached is used to store small chunks of objects or data resulted from database or API calls, or even from page rendering. It was adopted by YouTube to hold videos for serving user requests.

Many papers treat the cache replacement algorithms emphasizing their advantages and disadvantages. Two of them, widely used and implemented in our work, are the Least Frequently Used (LFU) and the Least Recently Used (LRU) [19]. In case of capacity miss when the cache has insufficient space to hold a new data block / file, LFU removes the least frequently used data block / file (that has the lowest use frequency) from the cache. The goal here is that lines that have been used many times are likely to be needed again. However, different from LFU, LRU removes the least recently used file (that has the longest time period since the time it was last used). In the LRU case, the principle is to evict the lines that were accessed the longest time ago, because according with temporal locality they are unlikely to be needed again very soon (although this is somewhat inconsistent with a Markovian probability that suggests to be preserved!).

In order to overcome some shortcomings of LRU and LFU, researchers from Intel, IBM and USA universities developed a framework competition – the Cache Replacement Championship (CRC) [20], targeting to find smarter solutions. In this section we briefly review three of this works and further describe some work dealing with video traffic and cache replacement algorithms for reducing data traffic in Social Media.

In [19] the authors proposed to the CRC a solution to enhance the Least Recently Used algorithm by keeping track of the relative use of each lines, in order to protect some frequently used lines.

In [21] is described a speculative cache replacement policy based on Instruction-based Reuse Distance Prediction. They quantify the temporal characteristics of the cache blocks at run-time by predicting the cache block reuse distances (measured in intervening cache accesses) exploiting the access patterns of instructions that belong to the cache block.

Jimenez [22] suggested another speculative policy of cache replacement driven by dead block prediction. The prediction follows the rule that blocks that were considered dead (were replaced before to be used again) when they last occupied the cache, will tend to be "dead" blocks the next time they occupy the cache. By not retaining these blocks in cache, is created additional space for useful blocks, with benefits on the hit rate. However, the solution drawback is represented by the high hardware requirements.

In [23] the authors introduced Pareto-based LFU (PLFU) and Pareto-based LRU (PLRU) and studied their performances in terms of miss count. The algorithms are based on Pareto principle (a.k.a. the 80/20 rule, Power laws, or Zipf's law [24]), keeping a percent of top popular (frequently accessed) videos of each video category and evict in case of miss the unpopular video. We implemented in our work their algorithms and also classical LRU and LFU and, taking into account the suggested example of "*Gangnam Style*" video from [13], we used an analytic formula that correlates the number of MB of data transferred through the Internet because we did not find

find videos in cache memory with the dissipated energy by Google servers.

3. Application Software Design

3.1 Hardware design of Pareto algorithms

presented in [25]. The Benchmark data is represented by number of views of each videos in consecutive weeks. The time period consists of number of weeks (20) we consider for a simulation.In Figure 1 we use Equation (1) for computing the metric I(k,i) for each week,as how many times the video *k* is accessed during week *i*, by decreas-

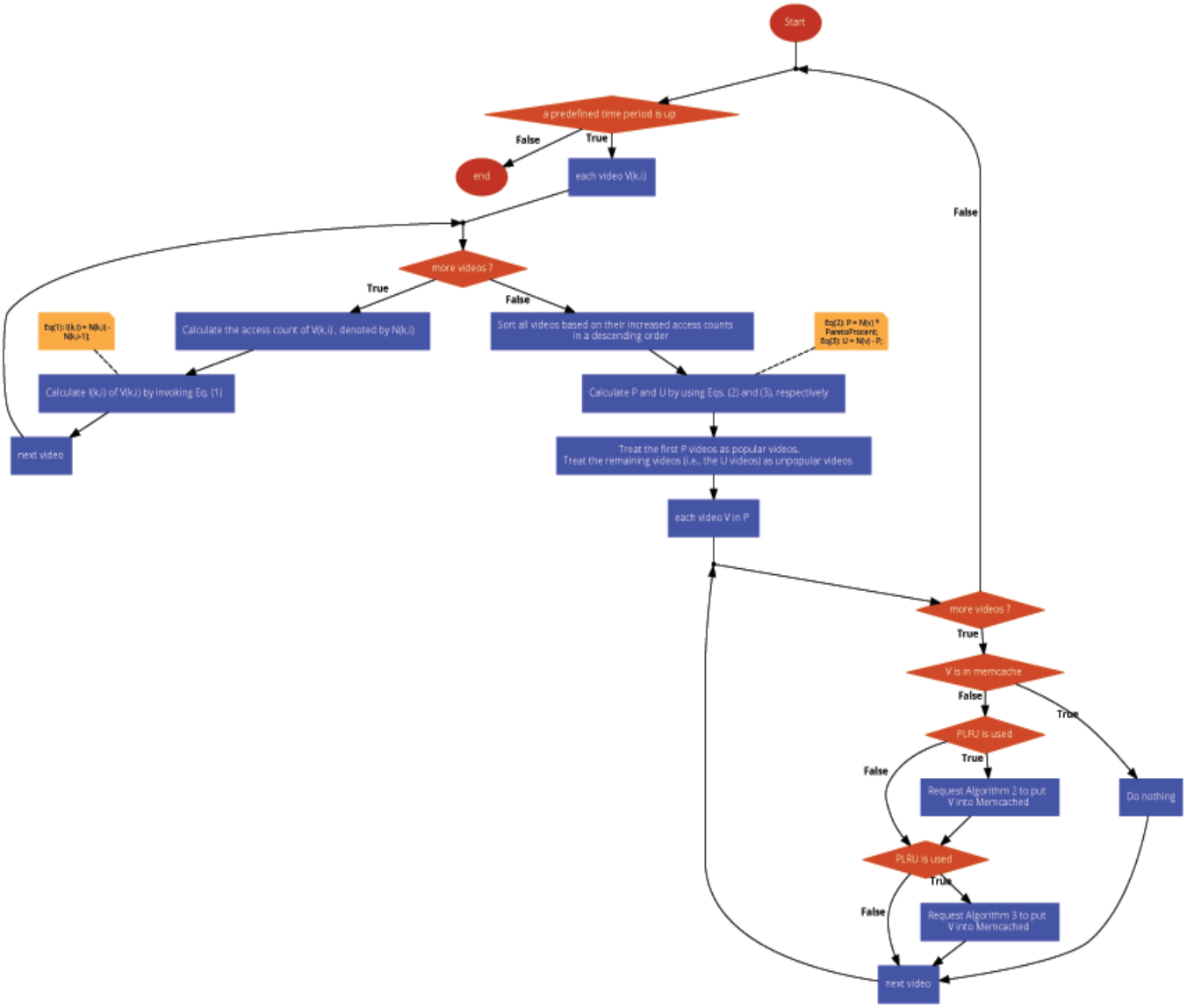

Figure 1. The general cache replacement algorithm: finding popular videos and applying PLFU and PLRU

For better understanding the mechanism of saving popular videos in MemCached and replacing the unpopular based on the two Pareto algorithms (PLRU and PLFU) we introduce the next three equations:

$$I(k,i) = N(k,i) - N(k,i-1) \quad (1)$$

$$P = N(v) \cdot ParetoPer\text{œ}nt \quad (2)$$

$$U = N(v) - P \quad (3)$$

The simulated YouTube datasets include many categories of videos (traces of accessed videos) collected between 16th of April 2008 and 27th of August 2008, and were

ing the video access count in week *i-1* (N(k,i-1)) from the video access count in week *i* (N(k,i)). Actually, I(k,i) is the increased access count of N(k,i). After computing the I(k,i) parameter for all videos these are sorted based on their increased access counts in a descending order in order to classify them in popular and less popular. In Equation (2) we varied the Pareto percent (Pareto Procent is 0, 10 and 20) from total number of videos stored in MemCached (N(v)) in order to determine the number of popular videos(P). The rest of them forms the unpopular videos (U in Equation(3)). After finding P and U, we simulate I(k,i) of all videos in a random order in our MemCached. We use PLFU or PLRU as replacing algorithms depending on the simulation configuration.

Regarding the MemCached size, today's servers can reach 24TB of RAM and 60 MB Cache [26, 27]. If we consider the average video size of 311.52 MB for a 4-minute-long 1080p video (see Table 4), we can store around 80000 videos in RAM. According to [28] 300 hours of video are uploaded every minute into YouTube databases and each day there are almost 5 billion views. Those shows the huge need for storage and effective video caching. In this approach, we considered a total capacity of 5000 videos. MemCached may keep any percent of these videos (usually in our tests 25%, 33% and 50% - values that represent a controlling factor of MemCached size). We consider a constant size for each video file, simplifying thus the replacement strategy against cases when videos have different capacities.Our 5000 benchmark videos generated over 21.3million views after first week.

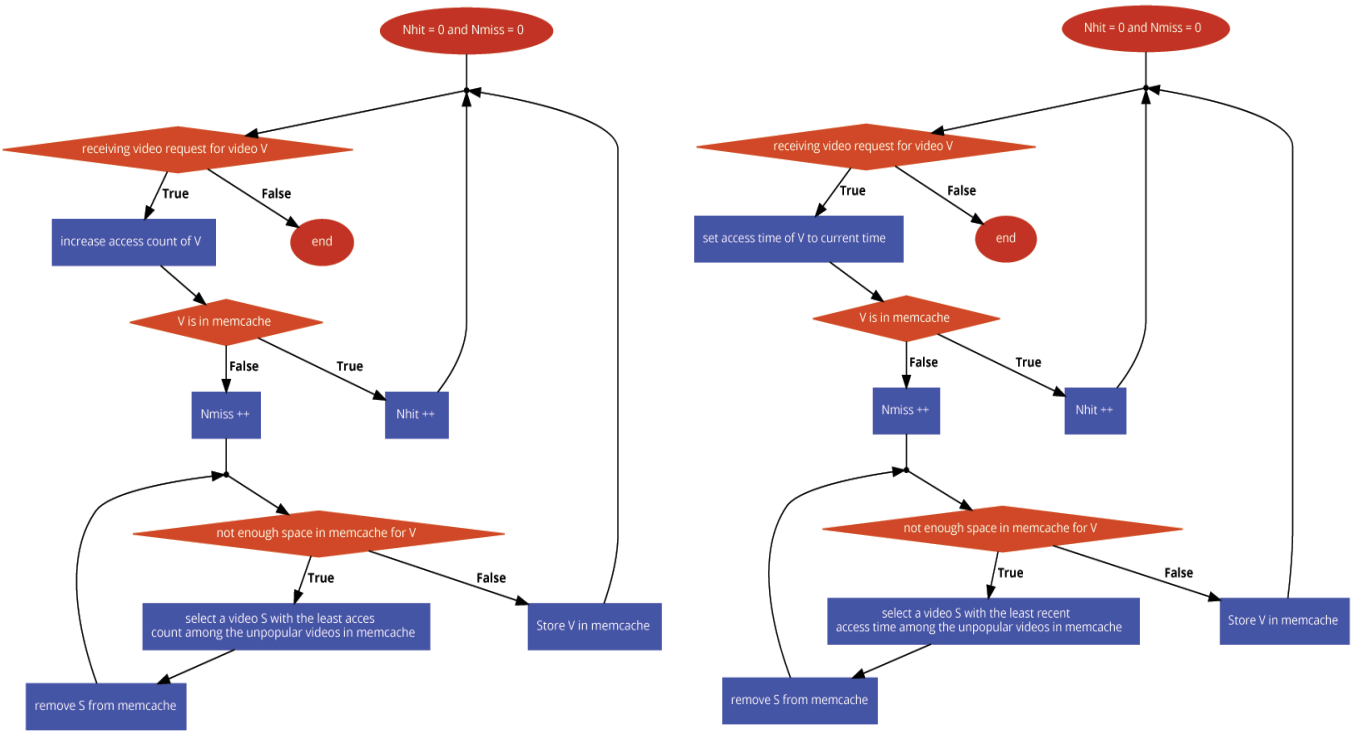

Figure 2. (left) PLFU replacement algorithms; (right) PLRU replacement algorithms

Input: The MemCached size and the sequence of video accesses during each week repeated for a whole period of 20 weeks;

Output: The number of hit/miss counts of videos in every week; the improvement in Hit rate according with different capacity of MemCached and Pareto percent will be correlated with saved energy of servers and finally with greenhouse gases emissions.

Procedure:

1: The Hit and Miss Counters are initialized with zero;
2: **while** (*receiving video request for each video V)*
3: Increment the access count of V
4: **if** (*V is now in MemCached)*
5: Increment the Hit Counter;
6: **else**
7: Increment the Miss Counter;
8: **if** (*is not enough space in MemCached to store V)*
9: Select a video S with the least access count (LFU) / or with the least recent access time (LRU) among the unpopular videos in MemCached
10. **do**
11: Remove the video S from MemCached;
12: **while**(*MemCached has not enough space to hold V);*
13: Save V in MemCached;

Figure 3. The PLFU/PLRU pseudocode

In Figure 2 (left) is illustrated how the PLFU algorithm works. Initially, hit and miss counters are set to 0. Then a sequence of video accesses is processed. Each video access is represented by the video identifier V. Then we increase the access count for video V. If video V is present in MemCached, we increase the Hit counter (Nhit) and then proceed to process the next request. Otherwise, the miss counter (Nmiss) is increased and if the MemCached has enough space to store this video, we store and continue with the next request. If there is not enough space, we select the least frequent used video (with the smallest access count) among the unpopular videos in MemCached and remove it to make space for the current requested video. After that we store the current video in MemCached and proceed with the next request.

In Figure 2 (right) is described the PLRU algorithm functionality. Initially, hit and miss counters are set to 0. Then a sequence of video accesses is processed. Each video access is represented by the video identifier V. Then we increase the access time for current video V. If video V is present in MemCached, we increase the Hit counter and then proceed to process the next request. Otherwise, the miss counter is increased and if the MemCached has enough space to store this video, we store and continue with the next request. If there is not enough space, we select the least recently accessed video (with the oldest access time) among the unpopular videos in MemCached and remove it to make space for the current requested video. After that we store the current video in MemCached and proceed with the next request.

For better understanding the workflow of PLRU and PLFU algorithms from figures 2, in figures 3 I briefly exhibited their pseudocode representations which are readjustments of those presented in [23].

3.2 The software architecture

The simulator is written in C# using Visual Studio 2015 Edition. The solution is composed of three projects:

• *Benchmark Loader*: responsible for parsing benchmark files and retrieving the data about YouTube videos.

• *Memcached System*: contains the implementation of the simulator, MemCached memory structure and the two replacing algorithms, PLFU and PLRU.

• *Youtube Simulator*: calls the previous two projects and combines their functionality.

Benchmark Loader project:

• Main class of this project is the *Benchmark Parser* static class and the method *Parse Directory*(). This method takes the following parameters:

• *directory Path*: the path on disk that points to the directory where the benchmark files are located.

• *number Of Files*: how many benchmark files to parse.

• *number Of Sample*: number of videos for which to extract the data.

Every file located at "{*directory Path*}/*trace* {*i*}. *txt*", where *i* goes from 1 to *number Of Files*, is parsed and its data is introduced into a dictionary of ***Youtube Video Data***. The convention is that benchmark file path should look like "{directory Path}/trace {i}. txt".

• ***Youtube Video Data*** class has two members, YouTube video identifier and a list with the views count of this video in each week.

• ***Log*** is a static class used for logging events, writing messages to console or to file.

Memcached System project:

• This project contains the abstract class **Memcached System**, a base class for **Memcached System LFU** and **Memcached System LRU** which implements the corresponding replacing algorithms,the **Simulator** and **Access Data** classes.

• **Access Data** class has two fields *Access Count* and *Replacement Data*, one used to keep track of the number of access counts of a video stored in MemCached and the other used when evicting a video from MemCached.

• The **Memcached System** class has two configurable parameters: *Pareto Procent* and *Maximum Number Of Videos* that MemCached can hold at a time. Other properties of are the hit and miss counters.

• *Pareto Procent parameter* range from 0 to 100, 0 meaning LRU or LFU replacement, while any other value means PLRU or PLFU replacement. It is used when computing the number of popular videos from MemCached that will not be replaced and the list with the video identifiers of unpopular videos that can be replaced.

• *Maximum Number Of Videos* is used to see if a new video that is not present in MemCached can be inserted or needs to replace another video.

• The abstract method *Access()* is implemented in the two subclasses **Memcached System LFU** and **Memcached System LRU**. Both implementations increase the hit count if the MemCached already contains that video. Otherwise it increases the miss count and checks if MemCached has free space. It adds the video to MemCached if there is space or, replace with other video otherwise. The video that is replaced is the video with the minimum *Replacement Data*. Then the *Access Count* for the current video is increased. The only difference between the two subclasses is how the *Replacement Data* is calculated. For the LRU subclass *Replacement Data* is *Access Count* while for the LFU is the order in which the video access came, the higher this value is the more recent it is.

• Other public methods are *Reset Global Counters()* used

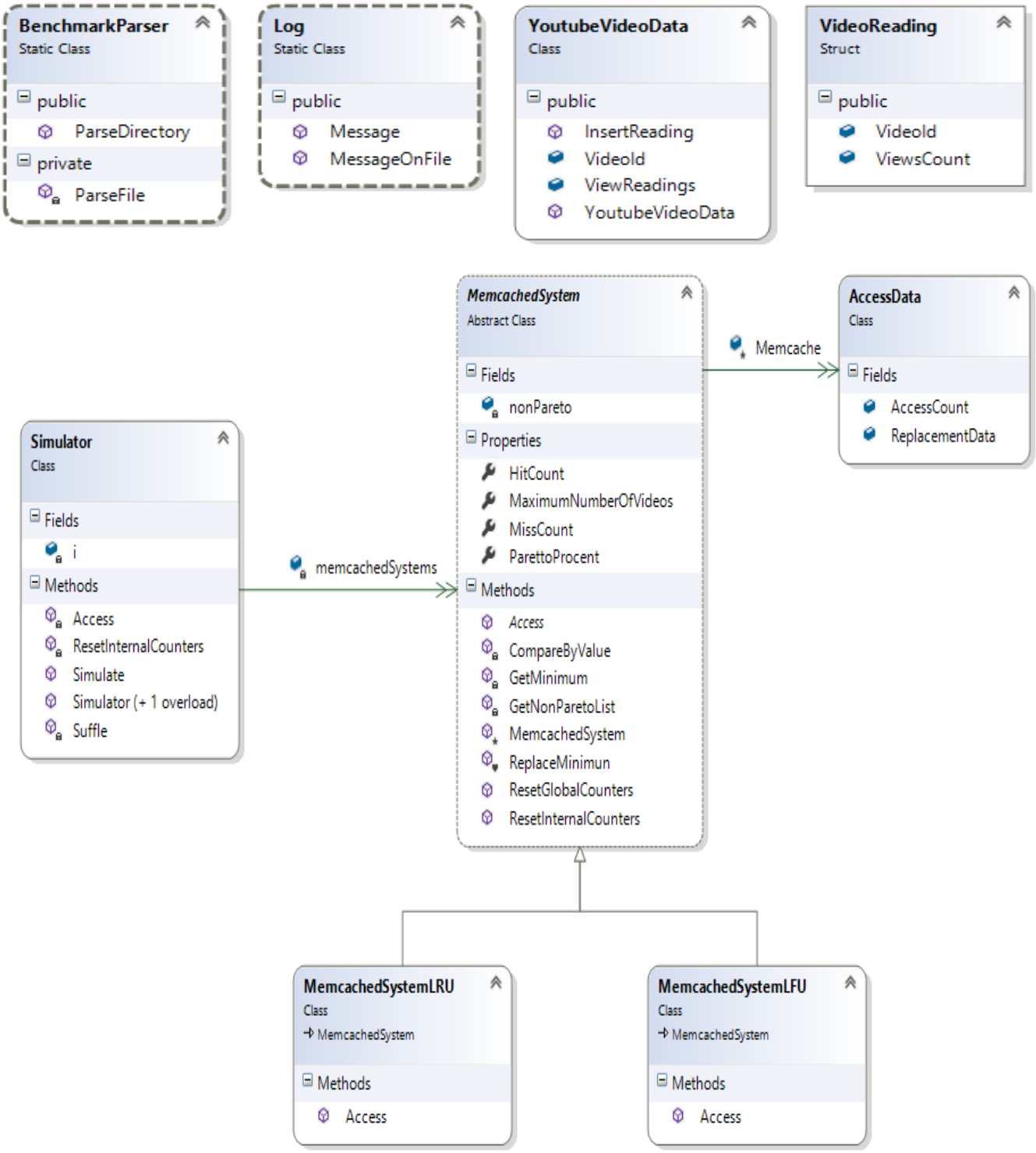

Figure 4. Application class diagram

used to reset the hit and miss counters and *Reset Internal Counters()* used to reset the *AccessCount* and *ReplacementData* of the videos present in MemCached and also to compute the list of unpopular videos.

• Other private or internal methods of **Memcached System** class are *Get Minimum()* that returns the video that will be replaced based on *Replacement Data* and *Replace Minimum()* that replaces the video with the smallest accessed video.

• The **Simulator** class takes an object or a list of **Memcached System** objects as parameter on its constructors. The single public method is *Simulate()* which takes a list of **Video Reading** objects. Then generates all the video accesses in a random order and simulates on each of the **MemCached** configuration in the *memcached Systems* list. After each simulation *Reset Internal Counters()* method is called on each **MemCached** configuration.

3.3 Application GUI

Before running, the user should configure the Pareto Cache simulator with the following parameters (see Figure 5):

• <*Number of traces (files)*> represents how many weeks / files will be considered (20 weeks in this case).

• <*Number of video samples*> stands for how many videos will be made simulations (5000 videos in this case).

• <*Memcached Size Ratio*> is a real number between 0 and 1, used to determine the MemCached capacity by multiplying this ratio with the number of videos (the parameter <*Number of video samples*>.

• <*Pareto Percent*>is an integer value between 0 and 99 representing the percentage of popular videos, actually the amount of videos from MemCached that will not be discharged.

• Select the folder that contains the trace files (the YouTube sites of video samples). Trace files should be named <*traceX.txt*> where X is the serial number (the analyzed week).In these files, each line contains the video's identifier (ID) and the number of visualizations at the time, separated by a tab character ('\t').

• After selection one of the two replacement algorithms (LRU or LFU) then, is added the new simulation configuration. After several configurations have been added, the user can start the simulation.

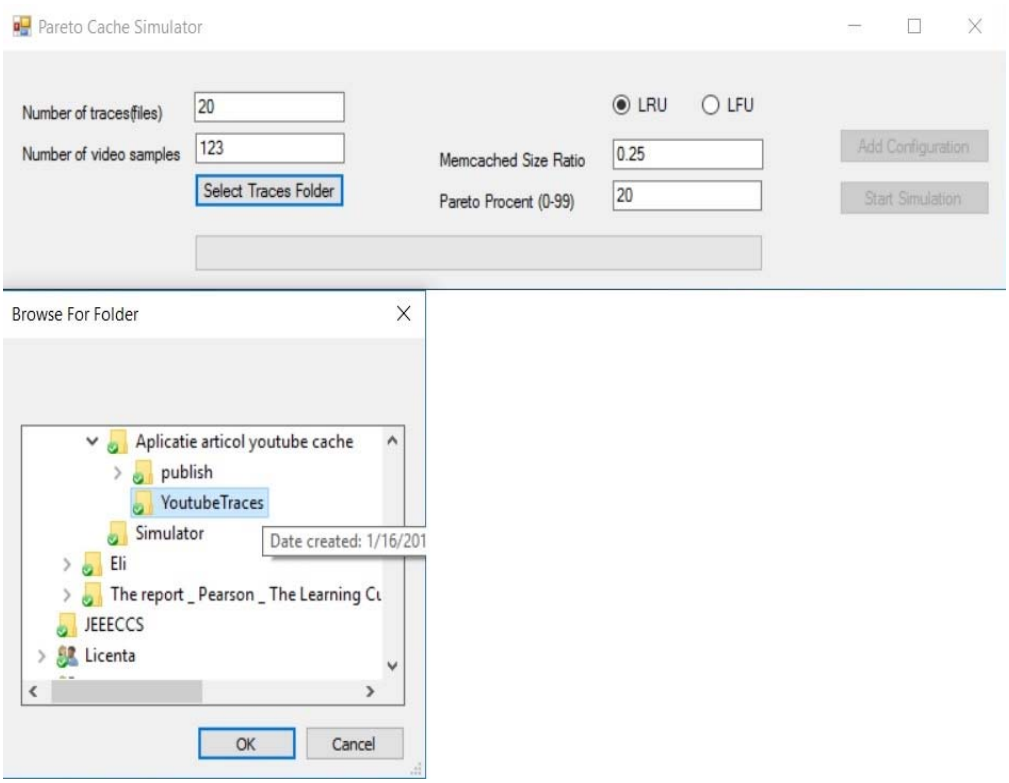

Figure 5. GUI: Setting the parameters and selecting the benchmarks folder

```
Number of videos 123
MemcachedLfuParetoProcent: 10  MaximumNumberOfVideos: 30
Week 1 : 2344476 495978;
Week 2 : 1956245 349864;
Week 3 : 1554644 286580;
Week 4 : 1434815 214261;
Week 5 : 1790062 187879;
Week 6 : 2251109 159972;
Week 7 : 1638474 179353;
Week 8 : 1369452 134109;
Week 9 : 1326626 128461;
Week 10 : 1334269 128154;
Week 11 : 1297650 122156;
```

Figure 6. Collecting simulation results in text form. MemCached keeps a quarter of videos

The simulation results file will appear in the application folder with the name <*OutputX.txt*> where X represents the current date of the computing system that run the simulator. The output file contains the following data (see Figure 6):

• <*Number of video samples*>

• <*MemCached capacity*>

• on each line, a MemCached configuration: *Pareto Percent* and replacement algorithm

• then, each line will be filled with data corresponding to each input (trace) file: The Hit and Miss counter specific for each configuration

selected again to fetch the traces again.

4. Simulation Results

Our first results emphasize the miss rate when is varying the MemCached sized. We have chosen three size of MemCached, one (large) that keeps half of all 5000 videos, one (medium) that keeps third of all 5000 videos and last one (small) that keeps a quarter of videos.

The main metrics obtained after simulation results are illustrated in table 1.

As the figures 7, 8 and 9 illustrate as larger the MemCached is as the miss count decreases. Also, the LFU algorithm provides better results than LRU and using

Performance Metric	Type (direct or indirect determined)	Measurement units
Miss rate	Direct	dimensionless
The numbers of misses reduced compared with a set reference	Indirect	dimensionless
The amount of data saved from transfer through the Internet, when videos are successfully retrieved from MemCached	Indirect	MB (Megabytes)
The amount of energy saved avoiding videos transfer through the Internet, when videos are successfully retrieved from MemCached	Indirect	GWh (Gigawatt Hour)
CO2 emissions avoided compared with a set reference	Indirect	Metric tones

Table 1. Dataset with the simulations' metrics

After each simulation, the user can introduce new configurations, overwriting the previous ones.If the user changes the parameter <*Number of video samples*> or <*Number of traces (files)*>, the folder location must be

a 20% Pareto percent (LFU 20) the missprediction is reduced about 26% (average gain obtained after all 20 weeks compared with simple LFU) no matter what size of MemCached is used.

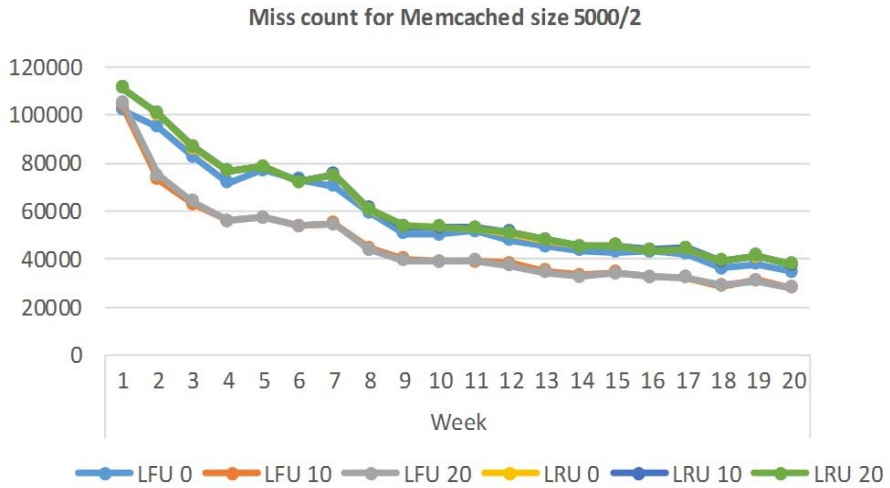

Figure 7. Miss count results if MemCached keeps half of videos

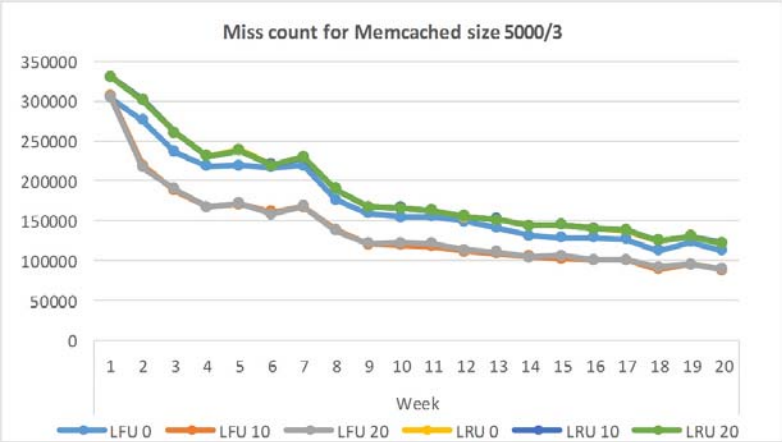

Figure 8. Miss count results if MemCached keeps third of videos

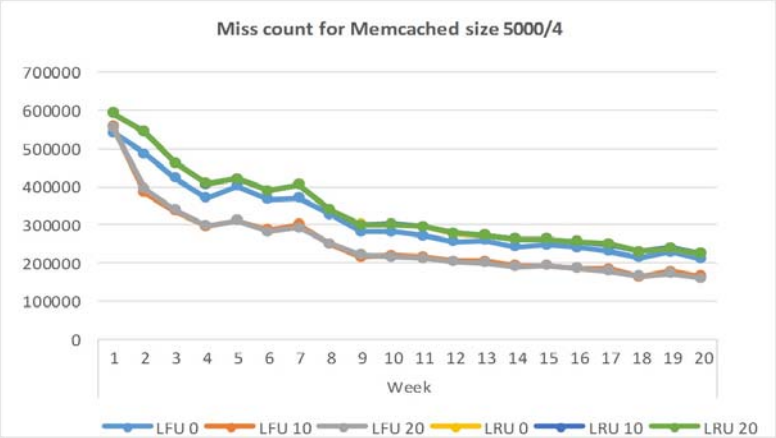

Figure 9. Miss count results if MemCached keeps a quarter of videos

For estimating the amount of MB saved from transfer through the Internet, when videos are successfully retrieved from MemCached, and, the energy saved by Google servers when the data transfer is reduced, we make the following considerations regarding the characteristics of video files according with YouTube recommendation for upload encoding settings [29], and energy required for each MB video transferred [13]:

Video:		
Type	Video Bitrate,Standard Frame Rate [Mbps]	Video Bitrate,High Frame Rate [Mbps]
	(24, 25, 30)	(48, 50, 60)
2160p (4k)	40	60
1440p (2k)	16	24
1080p (1k)	8	12
720p	5	7.5
480p	2.5	4
360p	1	1.5
Audio:		
Type	Audio Bitrate [Kbps]	
Mono	128	
Stereo	384	
Surround Sound (5.1)	512	

Table 2. Characteristics of video files

Bitrate	Video bitrate at 30 FPS[Mbps]	Video bitrate at 60 FPS[Mbps]
1080p	8.384	12.384
720p	5.384	7.884

Table 3. Bandwidth required for different Audio and Video bitrates

Average video play time[s]	Video Type	Size for a 4 minute video at 30 FPS[MB]	Size for a 4 minute video at 60 FPS[MB]	Average (30 FPS, 60 FPS) [MB]	Energy for 1MB transfer [kWh]	Total energy for 1 video transfer [kWh]
240	1080p	8.384 x 240 / 8 = 251.52	371.52	311.52	0.01	3.1152
	720p	5.384 x 240 / 8 = 161.52	236.52	199.02	0.01	1.9902

Table 4. Determining the total energy consumption for 1 video transfer

Memcahed size	Algorithm	Pareto Percent		Total energy for all video (1K) transferred [kWh]=Total number of misses x Total energy for 1 video (1k) transfer	Total [GWh]	Savings Compared to references [GWh]	%Savings Compared to references
5000/2	LFU	0	LFU 0	3611258.21	3.6113	0.1908	5.02%
		10	LFU 10	2862656.96	2.8627	0.9394	24.71%
		20	LFU 20	2861759.78	2.8618	0.9403	24.73%
	LRU	0	LRU 0	3787123.71	3.7871	0.0150	0.39%
		10	LRU 10	3802079.79	3.8021	0	0.00%
		20	LRU 20	3799575.17	3.7996	0.0025	0.07%

Table 5. Determining thetotal energy consumed for transferring all video that miss and the saved energy in case of using large MemCached and comparing with the worst replacement algorithm – LRU 10

In our experiments we have considered Stereo audio for all videos, an average video length of 4 minutes and HD (Standard Frame Rate) and Full HD (High Frame Rate) formats as average. We present results for two bit rate videos (1080p and 720p).

Taking into account the data analytics considerations used in [13] we used a formula that correlates the number of MB of data transferred through the Internet, because we did not found videos in MemCached, with the dissipated energy by Google servers responsible further for CO2 footprint. The amount of energy used by Google to transfer 1 MB across the Internet [13] is 0.01 kWh (a rough average), and displaying it uses 0.002 kWh (depending on the destination device). We neglect in our experiments the energy required to display video. Table 4 exhibits the total energy consumption for transferring 1 video of 4 minutes average length for two types of videos (Full HD and HD resolutions). We repeated the computations for formerly features of video statistics dated from 2008 [25] and we obtained 0.084 kWh energy consumption for 1 video transferred having an average video size of 8.4 MB (37 times smaller than the actual size of Full HD videos which is 311.52 MB).

Table 5 presents the total energy consumption (after 20 weeks) for transferring through the Internet all Full HD (1080p) videos that generated miss in the largest MemCached tested. We consider LRU 10 as reference (highlighted with gray in tables) as it is the worst replacement algorithms from all implemented. The table also illustrates the saved energy provided by replacement algorithms comparing with reference. With a Pareto percent of 20, the LFU algorithm (LFU 20) reduced the energy with almost 940 MWh (comparing with LRU 10), being with 24.73% more *green*.

The tables 6 and 7 repeated the statistics from Table 5 for

different size of MemCached (medium and small). For Table 7 the worst replacement algorithms became LRU 0 such that it became the reference.

As it can be observed comparatively studying the tables 5, 6 and 7, the savings energy versus reference algorithms in MemCached, compared with reference.The best energy saved of 5.33 GWh contributes to avoiding of 3,751 metric tones of CO2 emissions, which is equivalent with greenhouse gas emissions during one year from 792 passenger vehicles driven or from electricity use of 554 homes.

MemCached Size	Algorithm	Pareto percent		Total energy for all video (1k) transferred [kWh]= Total number of misses x Total energy for 1 video (1k) transfer	Total[GWh]	Savings compared to reference [GWh]	% Savings compared to reference
5000/3	LFU	0	LFU 0	10870705.35	10.8707	0.8091	6.93%
		10	LFU 10	8654798.17	8.6548	3.0250	25.90%
		20	LFU 20	8693629.138	8.6936	2.9862	25.57%
	LRU	0	LRU 0	11666330.54	11.6663	0.0135	0.12%
		10	**LRU 10**	11679816.24	11.6798	0.0000	0.00%
		20	LRU 20	11670099.94	11.6701	0.0097	0.08%

Table 6. Determining thetotal energy for transferring all video that miss and the saved energy in case of using medium MemCached and comparing with the worst replacement algorithm – LRU 10

MemCached Size	Algorithm	Pareto percent		Total energy for all video (1k) transferred [kWh]= Total number of misses x Total energy for 1 video (1k) transfer	Total[GWh]	Savings compared to reference [GWh]	% Savings compared to reference
5000/4	LFU	0	LFU 0	19435626.88	19.4356	1.5040	12.88%
		10	LFU 10	15669237.94	15.6692	5.2704	45.12%
		20	LFU 20	15602859.25	15.6029	5.3368	45.69%
	LRU	0	**LRU 0**	20939651.67	20.9397	0.0000	0.00%
		10	LRU 10	20932866.77	20.9329	0.0068	0.06%
		20	LRU 20	20934181.38	20.9342	0.0055	0.05%

Table 7. Determining thetotal energy for transferring all video that miss and the saved energy in case of using small MemCached and comparing with the worst replacement algorithm – LRU 0

increase as the MemCached size decreases. The best algorithm, both from miss rate and from saved energy viewpoint, proved to be Pareto LFU algorithm. In case of small size of MemCached (5000/4) and using PLFU 20, the energy saved compared with reference (LRU) is 45.69%, meaning 5.33 GWh quite significant. Starting from the Greenhouse Gas Equivalencies Calculator [30] which considers emission factor of 7.03×10^{-4} metric tones CO2 / kWh for converting reductions of kilowatt-hours into avoided units of carbon dioxide emissions, we compute in Table 8 how much CO2 emissions are bypassed if we apply Pareto-based algorithms for video files replacement

5. Conclusions And Future Work

This paper presents a different teaching approach of cache memory concept in the microprocessors systems field starting from societal challenges such as the huge need for storage and effective video caching, with a negative impact on the energy consumption of servers from data centers and on the emission of greenhouse gases that finally causes climate changes.The interesting fact emphasized by our work is that, starting from some evaluations of Pareto-based algorithms for cache blocks replacement, and, studying temporal locality of YouTube videos,

MemCached Size	Algorithm	Pareto percent		Savings compared to **reference**[GWh]	CO2 emissions avoided versus **reference**[Metric tones]	Greenhouse gas emissions from:	
						Passenger vehicles driven for one year	Homes' electricity use for one year
5000/4	LFU	0	LFU 0	1.5040	1,057	223	156
		10	LFU 10	5.2704	3,704	782	547
		20	LFU 20	5.3368	**3,751**	**792**	**554**
	LRU	0	**LRU 0**	0.0000	0	0	0
		10	LRU 10	0.0068	4,8	1	0.706
		20	LRU 20	0.0055	3,9	0.816	0.571

Table 8. Determining the avoided CO2 emissions in case of using small MemCached and comparing with the worst replacement algorithm – LRU 0

it shows that algorithm's performance positively influences the cache hit rate, reducing the number of data transfers through Internet, reducing the data center energy and finally reducing CO2 emissions ensuring a cleaner environment. We software implemented a Pareto Cache simulator applied for YouTube videos varying the percent of popular video and MemCached capacity. The simulators outputs are: the miss rate, the energy consumption and the greenhouse gas emissions produced by servers for transferring data through Internet in case of miss.

From our results it can be seen that there is a small difference between keeping of 10% popular videos and keeping of 20%.Miss rates are going up fast as we decrease MemCached size. The best results obtained are with Pareto LFU. With 20% popular videos, the miss rate decrease from 0.3% at a capacity factor of 1/2 (large) to 1.68% at 1/4 (small), being close to a six-time increase. Statistically, energy consumed on video retrieval from backing storage is reduced by over 45.69%. As expected, having a small cache implies more cache misses and bigger energy consumption on bringing the data from backing storage. Judging from perspective of energy savings compared with the worst algorithm and not necessarily strictly from energy consumption viewpoint, we noticed superiority of PLFU over the rest of implemented algorithms. Thus, the most viable configuration we benchmarked has 1/4 MemCached capacity factor and Pareto percent is 20%, because in this case we save the highest amount of energy and CO2 emissions.

Today's mainstream resolution is 1080p (1k) and the size of a 4minutes long video is around 311.52 MB. Using this up to date video size, the saved energy on our benchmarks are 5.33 GWh on 5000 videos over 20 weeks. These figure can go up as many smartphone cameras now support 4k filming and 4k video quality is becoming the new mainstream. Furthermore, from 2015 YouTube supports 8k videos [31]. Even though 4k is not mainstream yet, on CES 2017 event, Dell announced an 8K monitor [32].This shows us that any small energy saving is important, even more since, according to [10], the global computing requirements (software applications and hardware devices) might consume more energy than the world can produce by 2040.

In conclusion, in a modern society based on comfort and large scale mobile communications with massive data, it is mandatory to find, design and use *Green IT* solutions at every level (hardware, software), including in the teaching process of young generation.

As further work, we try to increase the user satisfaction by computing how much video-retrieval time per week for YouTube can be reduced by the advanced cache replacement algorithms. Also, teaching of other computers science concepts (Database and E-commerce, Big Data processing, etc.), with straight connection to societal challenges is one of our future research direction.

Acknowledgements

I would like to thank to my student Alexandru Matei from Computer Science and Electrical Engineering Department for providing useful technical help in the software implementation of cache replacement algorithms.

References

[1] Florea, A., Klein, A., Badea, V., Stefanescu, M., Gellert, A. (2013). *Using FOCAP Tool for Teaching Microarchitecture Simulation and Optimization*, *In*: Proceedings of the 17th International Conference on System Theory, Control and Computing, Sinaia, 11-13th October 2013, p. 225-230.

[2] Florea, A., Ratiu, A., Gellert, A., Vintan, L. (2011). *A Visual Simulation Framework for Simultaneous Multithreading Architectures*, *In*: Proceedings of the 25th

European Conference on Modeling and Simulation (ECMS 2011), Krakow, Poland, June 7-10, 2011, p. 403-409.

[3] Florea, A., Radu, C., Calborean, H., Crapciu, A., Gellert, A. (2007). An Interactive Graphical Trace-Driven Simulator for Teaching Branch Prediction in Computer Architecture, *In*: The 6th EUROSIM Congress on Modelling and Simulation (EUROSIM 2007), p. 58 (1-7), September 9-13, Ljubljana, Slovenia 2007, special session: Education in Simulation / Simulation in Education I.

[4] Ericsson Energy and Carbon Report (2013). On the impact of the Networked Society, June, 2013, https://www.ericsson.com/res/docs/2013/ericsson-energy-and-carbon-report.pdf, accessed 7 January 2017.

[5] Duranton M. et. al (2017). The HIPEAC Vision, 2010, https://www.hipeac.net/assets/public/publications/vision/hipeac-vision-2010.pdf, accessed 7 January 2017.

[6] Ericsson Energy and Carbon Report (2014). Including Results from the First ever National Assessment of the Environmental Impact of ICT, November, 2014, https://www.ericsson.com/assets/local/about-ericsson/sustainability-and-corporate-responsibility/documents/ericsson-energy-and-carbon-report.pdf, accessed 14 January 2017.

[7] Calero, C., Piattini, M. (2015). *Green in Software Engineering*, Springer, 2015.

[8] Make IT Green: Cloud Computing and its Contribution to Climate Change, Greenpeace Report, March 2010, Greenpeace International, http://www.greenpeace.org/international/Global/international/planet-2/report/2010/3/make-it-green-cloud-computing.pdf, accessed 8March 2017.

[9] Grosclaude, J-Y., Pachauri, R.K., Tubiana L. (2014). *Innovation for Sustainable Development*, The Energy and Resources Institute, February, 2014.

[10] Duranton, M., De Bosschere, Koen., Gamrat, Christian., Maebe, Jonas., Munk, Harm., Zendra, Olivier (2017). *The HIPEAC Vision*, 2017, https://www.hipeac.net/v17, accessed 8 March 2017.

[11] Opoczky, M.K. (2016). *Shaping the future of offshore wind*, *Research EU results magazine,* Issue 57, European Union Publishing Hall, November, 2016.

[12] Ia-Manee, N., Sophatsathit, P., *(2013)*. Reducing Energy Consumption in Programs Using Cohesion Technique, *International Journal of Computer Theory and Engineering*, Vol. 5 (4) 621-625.

[13] Procaccianti, G., Fernández, H., Lago, P. (2016). *Empirical evaluation of two best practices for energy-efficient software development*, *Journal of Systems and Software* 117. 185-198.

[14] Kwon, Y. W., Tilevich E. (2013). *Reducing the Energy Consumption of Mobile Applications behind the Scenes*, *In:* Proceedings of the 29th IEEE International Conference of Software Maintenance, p. 170-179, September, 2013.

[15] *Webster's New World College Dictionary*, http://websters.yourdictionary.com/, accessed 2 January 2017.

[16] Gregersen, E., Sir Maurice Vincent Wilkes, British computer scientist, Encyclopædia Britannica, https://www.britannica.com/biography/Maurice-Vincent-Wilkes, October, 2015, accessed 7January 2017.

[17] *IBM System/360 Model 85*, https://en.wikipedia.org/wiki/IBM_System/360_Model_85, accessed 2January 2017.

[18] *MemCached*, http://en.wikipedia.org/wiki/Memcached, accessed 10 January 2017.

[19] Peress, Y., Finlayson, I., Tyson, G., Whalley, D.,CRC: Protected LRU Algorithm. *In*: JWAC 2010-1st JILP Workshop on Computer Architecture Competitions: cache replacement Championship, June, 2010.

[20] 1st JILP Workshop on Computer Architecture Competitions (2017). (JWAC-1):Cache Replacement Championship Framework, http://www.jilp.org/jwac-1/¸ accessed 2 January 2017.

[21] Petoumenos, P., Keramidas, G., Kaxiras S. (2010). Instruction-based reuse distance prediction replacement policy, *In*: JWAC 2010-1st JILP Workshop on Computer Architecture Competitions: cache replacement Championship,June, 2010.

[22] Jimenez, D.A.(2010). Dead block replacement and bypass with a sampling predictor, *In*: JWAC 2010-1st JILP Workshop on Computer Architecture Competitions: cache replacement Championship, June, 2010.

[23] Lee, M., Leu F.Y., Chen Y.P. (2014). Cache Replacement Algorithms for YouTube, *In*: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, p. 743-750, IEEE, 2014.

[24] Newman, M.E.J. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics* 46, 323–351.

[25] Cheng, X., Dale, C., Liu, J. (2008). *Statistics and Social Networking of YouTube Videos*, *In*: Proceeding of the 16th IEEE International Workshop on Quality of Service, p. 229–238 (2008).

[26] https://www.bios-it.co.uk/News/2016/06/Introducing-the-worlds-highest-density-8-way-Supermicro-Server, accessed 6 March 2017.

[27] https://www.supermicro.com/products/system/7U/7088/SYS-7088B-TR4FT.cfm , accessed 6 March 2017.

[28] *You Tube Company Statistics (2017).* http://www.statistic-brain.com/youtube-statistics, accessed 6 March 2017.

[29] *YouTube recommended upload encoding settings*, https://support.google.com/youtube/answer/1722171?hl=en, accessed 25 February 2017.

[30] United States Environmental Protection Agency, Energy and the Environment, *Greenhouse Gases Equivalencies Calculator - Calculations and References*, https://www.epa.gov/energy/greenhouse-gases-

equivalencies-calculator-calculations-and-references, accessed 8 March 2017.

[31] Maxwell, T. (2015). *Videos supporting 8K resolution starting to appear on YouTube*, June, 2015, https://9to5google.com/2015/06/08/videos-supporting-8k-resolution-starting-to-appear-on-youtube, accessed 8 March 2017.

[32] Kessler, D. (2017). *Dell's new 8K monitor is simply jaw-dropping*, Dell Consumers Electronics Show (CES 2017), Las Vegas, January, 2017, http://www.windowscentral.com/dell-8k-monitor, accessed 8 March 2017.