Neves *et al. Journal of the Brazilian Computer Society* (2015) 21:8
DOI 10.1186/s13173-015-0032-1

Journal of the Brazilian Computer Society
a SpringerOpen Journal

RESEARCH Open Access

A novel caching algorithm for VoD proxy implementation and its evaluation including a new set of metrics for efficiency analysis

Bruno S Neves[1,2*], Anderson S Venturini[2] and Altamiro A Susin[1]

Abstract

Background: Video on demand (VoD) is a fast-growing digital service that requires a substantial amount of hardware resources for its implementation. To reduce the costs of running this service, an alternative is to use proxies that cache the most important portions of the video collection in order to meet the demand for this content in place of the primary server of the VoD system.

Methods: To improve the efficiency of the proxies, we proposed a novel caching algorithm that explores the positioning of the active clients to determine the density of clients inside a time window existing in front of each video chunk. The algorithm attributes a higher caching priority to the video chunks with greater aggregate density in the memory. To evaluate our approach, we compared it with others of similar nature using both traditional metrics like hit ratio, as well as physical metrics such as the use of processing resources. This complementary set of metrics is produced by our simulator that, as far as we know, is the first one of its kind to enable the evaluation of hardware consumption used to implement the VoD proxy.

Results: Results show that the novel algorithm can achieve a higher hit ratio while requiring a little more effort from the hardware. Additionally, it was identified that the processor constitutes the major bottleneck to this application when demand increases.

Conclusions: Among the recent emergence of caching strategies which consider the positioning of clients as criterion for caching, the strategy of prioritizing the chunks with greater density of previous clients showed to be a more efficient solution.

Keywords: Video; Proxy; Caching; Algorithm; Scalability; Hardware

Background

Recent estimates indicate that video traffic, including but not limited to digital TV, video on demand (VoD), and non-real-time transmissions, like media sharing, will account for 86 % of all global data traffic over the network by 2016, and video on demand will account for 54 % of this traffic [1]. The same source estimates that the VoD traffic will triple in the same period. This suggests that VoD streaming will require a more suitable support for its operation; otherwise, the cost to expand the video traffic will be challenging.

To reduce the consumption of network bandwidth while at the same time providing an improved quality of service, an alternative was created by utilizing proxies near the local networks of the clients. Each one of the proxies acts as a secondary server caching the most accessed information in order to provide this information to its clients in place of the main video server, as shown in Fig. 1. Therefore, the proxy efficiency is fundamental in determining the total cost and productivity (number of simultaneous clients served by the system) to the VoD service.

Based on this, the most important aspects for the efficiency of the VoD proxy is the caching algorithm [2, 3] (also referred to as cache replacement algorithm) used to determine the preferred content to be kept in the cache (memory of the proxy). The best content to store in the

*Correspondence: brunoneves@unipampa.edu.br
[1] Federal University of Rio Grande do Sul, Osvaldo Aranha Av., 103, 90035-190 Porto Alegre, Brazil
[2] Federal University of Pampa, Lane 45, 1650, Malafaia, 96413-170 Bagé, Brazil

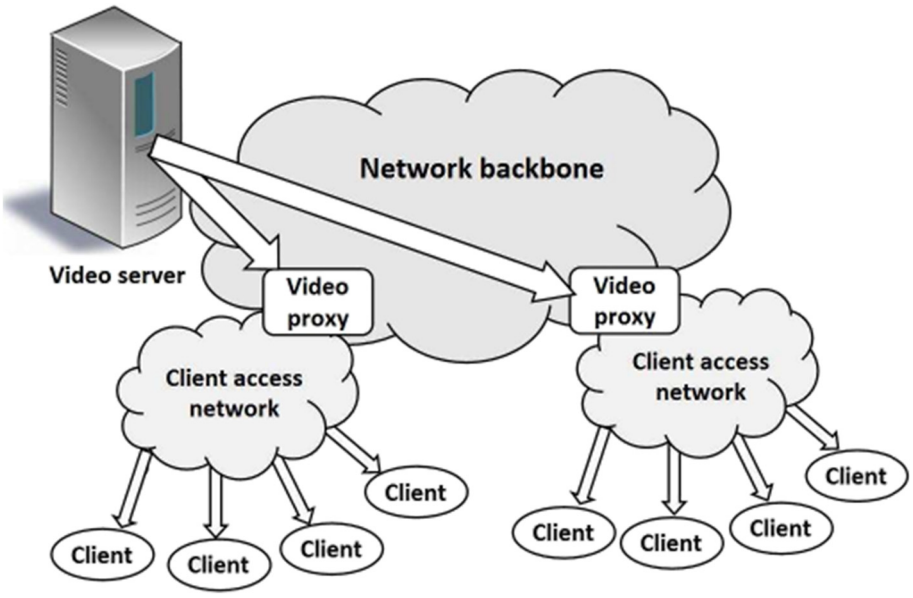

Fig. 1 Strategy for using VoD proxies. Proxies serve the VoD clients by reducing the load on the main server and the network backbone

cache is that that will allow the proxy to supply the greatest number of requests generated by the clients of the VoD system.

A similar problem is addressed by replication algorithms [3] that create replicas of the content available on the main server and store them in surrogate servers. However, contrary to caching algorithms, replication algorithms determine which objects should be replicated and in which servers the replicas must be carried out taking into account load balancing between the surrogate servers as the most important decision factor [4, 5].

We have outlined this difference between the two classes of algorithms to clearly define the scope of the research which focuses on the design and analysis of video caching algorithms employed in VoD proxies. Additionally, the main target of our study is a centralized approach[1] where it is assumed that the decisions made by a caching algorithm running on a VoD proxy are not based on any kind of collaboration with other proxies in the system.

Many caching algorithms use the history of the access pattern of the clients as a reference to try to predict which portions of the collection will be needed in the future [6–11]. Thus, the most accessed video objects should be stored in the cache in order to easily serve future requests. However, as this prediction begins to fail, the proxy efficiency is decreased and the caching algorithm needs to update the frequency history in order to define a new target content to be stored for the next period.

An example of this caching approach is proposed by Li et al. [12] who presents a caching model based on two popularity control variables: the first variable counts the number of accesses to a video object in a short-term interval (from current to past time) and the second variable counts this number in a long-term interval. If the number of accesses observed in any of these intervals exceeds the threshold value for its respective interval, the item is marked with a higher caching priority; otherwise, the item receives a lower caching priority.

The concern of Li et al. [12] in analyzing the relevance of video objects at different moments brings out a broader reflection around the dynamicity relative to the changes in the popularity of the video fragments. However, from our point of view, the main deficiency of the Li et al.'s approach [12] is the positioning of the analysis intervals used by their algorithm since both temporal windows are located from the present to the past. As a consequence, the algorithm is subjected to making imprecise caching decisions given that the same variations in the popularity analyzed by it can also cause the actual demand to be different from the one predicted by its estimation variables.

Chen et al. [13] based the video popularity function of their algorithm on two main factors: duration and recency of the last access to each item of the collection. Moreover, they proposed that each video occupies a portion of the memory in proportion to its respective popularity. In our opinion, the algorithm presents unexplored aspects that can potentially improve the proxy efficiency. For example, the algorithm could get an increase in hit ratio if the duration of the accesses measured by it could be correlated to the video segments accessed during this period. Thus, the algorithm could determine which parts of a video are usually more requested by the clients and, consequently, the less requested parts could be discarded to reduce memory consumption.

Ishikawa and Amorim [14] presented a caching algorithm named Collapsed Cooperative Video Caching (CCVC), inspired by the caching policies used in peer-to-peer networks, that works on two levels. On the first level, the algorithm selects a video with smaller popularity (lower number of active[2] clients) as a victim. On the second level, the algorithm decides which parts of the video selected on the first level should be discarded in order to free up space for allocation of the content received from the main server.

In addition, a key aspect of the approach used by Ishikawa and Amorim is to assign a higher caching priority for video blocks allocated between two or more clients who watch the same video. Consequently, the video blocks brought to the memory to serve the older clients of the system tend to remain in the memory to be accessed by the more recent clients who watch the same video. This feature makes CCVC one of the first algorithms to take into account the active clients who have not yet accessed certain portions of a video as a factor in determining which contents should be cached.

More recently, Hong et al. [15] proposed an algorithm (named CC) that, compared with the CCVC, employs, in more clear way, a new paradigm (which we refer to as *Look Ahead*) to decide which content should be cached. The algorithm takes into account the current positioning of the clients who are watching a particular item of the collection as a way to determine the effective number of hits in the future for each video segment of this item. Only clients who have not yet accessed a particular segment count for the number of future accesses to that segment.

Nevertheless, in our view, one flawed aspect of this approach relates to the fact that the lack of a limit with respect to the temporal distance between the watching positions of the clients and the segment causes the segments at the end of the videos to have a greater number of future requests associated to them, thus remaining in the memory of the proxy although there may be segments at the beginning of the video with higher demand associated to them in the short term.

Finally, Wu et al. [16] proposed a system that estimates the time it takes to occur the access to each video segment. The most important aspect of this strategy is measuring the distance between each segment and the closest client that will access to this segment. When a client closer to a segment performs an access to it, the system automatically recalculates the distance between the segment and the next closest client.

This strategy, however, shares the same problem present in Hong's approach, since by prioritizing the preservation in memory of the segments that will be accessed soon not necessarily the segments with higher demand will be preserved, because clients' concentration can be larger near the segment whose distance to the closest client is also greater.

Methods

According to what was presented in the previous section, the *Look Ahead* algorithm type, such as [14–16], bases its caching decisions on actual client requests, therefore establishing a more solid baseline in determining which portions of each video must be kept in cache in order to provide greater efficiency to the proxy when the number of concurrent clients is high.

However, we discovered a way to improve the caching logic employed by the previous algorithms of this type, and based on it, we proposed a novel and more efficient caching strategy. Our strategy takes into account the density of clients existing in a bounded region near each chunk thus preventing the clients that are distant of a given chunk to interfere in the calculation of the caching priority for that chunk.

To evaluate our algorithm, we developed a simulator that is not only able to analyze the efficiency using conventional metrics (such as the hit ratio), but also is able to evaluate the use of computational resources of the underlying hardware. Therefore, the simulator enables us to determine how the hardware is able to restrict the system scalability taking into consideration scenarios where the workload is high. As far as we know, this ability makes our simulator the first one to enable the analysis of the proxy performance from an architectural point of view.

Briefly summarizing, this method is based on three main objectives:

1. The design of a novel and more scalable video caching algorithm for use in high workload scenarios of the VoD service.
2. The development of a simulator capable of identifying the bottlenecks of the subjacent architecture of the VoD proxy.
3. A deeper comparative evaluation of scalability provided by different caching algorithms.

The remainder of this section is organized as follows: the "Proposed algorithm" subsection describes the caching algorithm we propose in this work, while the "Experimental environment" subsection presents the simulator built to assess the performance of our algorithm.

Proposed algorithm

The main idea of our caching strategy came from a broader observation of all scalability aspects to maximize not only the efficiency related to the network bandwidth but also other physical resources of the proxy which are often exposed to peaks in the workload to which the system needs to respond appropriately.

The differentials of our approach arise from the understanding that, in high-demand scenarios, the proxy should make the best choice with respect to the use of its resources by considering only the actual demand (created exclusively by the clients who have already started their video section in that system) existing inside a short-term time window of a few seconds[3] in the direction of the future. Otherwise, the use of these resources aiming at the service qualification for the longer-term demands (possible clients) could reduce the proxy scalability to support the current demand.

According to this view, when the proxy works to serve potential long-term requests (as occurs when caching prefixes), the cache content exchange rate tends to be smaller since under these conditions the caching philosophy does not prioritize current load conditions. As a consequence, the bandwidth associated to all the proxy devices tends to be underutilized since the system performs no data flow or any other kind of computation to update the cache content aiming to sustain the best scalability possible for the proxy and, thus, meet the current load.

Conversely, according to our proposal, under high workload conditions, the caching algorithm should constantly be organizing the data allocation flow in order to maximize the proxy productivity. For this purpose, it should evaluate and select what video chunks have the best demands, i.e., higher number of clients who will access them in short term, prioritizing the data allocation for this content.

By implementing a time window to determine the density of clients (number of clients per time interval) in front of each video chunk, our algorithm is able to work more actively to monitor and update the system variables according to each load state, making most appropriate use of the available hardware.

The sub-subsections below describe the Current demAnd Rather Than futurE (CARTE) algorithm we developed to evaluate the impacts on the VoD proxy performance generated by the use of the design features previously mentioned. To review, they are as follows:

- Monitoring the density of clients in future short term.
- Reacting to the current load state conditions by performing memory content updates.
- Employing all available hardware bandwidths to increase the scalability of the proxy.

The "Video organization" sub-subsection shows how the algorithm logically divides the videos into smaller chunks and discusses the basics of the caching priority binding for each kind of chunk. The "Caching logic" sub-subsection explains in more detail the caching decision logic employed by CARTE, presenting mathematical descriptions on how our algorithm works.

Video organization

The strategy employed by CARTE to divide the videos in smaller chunks, called *Sequences*, is inspired by the *Linking Slots Management Police* used by the CCVC algorithm [14]. As we demonstrate below, the organization of video into sequences allows a more efficient use of the available memory resources in comparison to the widely used classical division of the videos into segments of equal size. A similar reasoning could be utilized to compare the use of sequences to the use of variable length segments, such as those proposed in [17].

Figure 2 illustrates the video organization process used by CARTE where each square represents a video block and the horizontal lines of successive squares (there are three in the figure) correspond to the same video clip. Each row from top to bottom describes the state of the blocks belonging to the video clip 1 s further along in time.

Squares containing a white circle represent the video blocks that are not present in the memory of the proxy at that current time. Each white square indicates that one or more clients are receiving the correspondent video block. Blocks that are being read cannot be discarded by the replacement algorithm when free memory becomes scarce. Hatched squares indicate a trail of blocks left in the memory by a client as it progresses every second (assuming constant bit rate (CBR) transmission) to the next video block. Hatched blocks represent the preferred content for discarding when the free memory is scarce.

When the reading position of a client limits a trail to the left, the algorithm increases the caching priority of the blocks belonging to the trail to an intermediate value (indicated by triangles within the square). Since these blocks tend to be utilized by the clients who are positioned

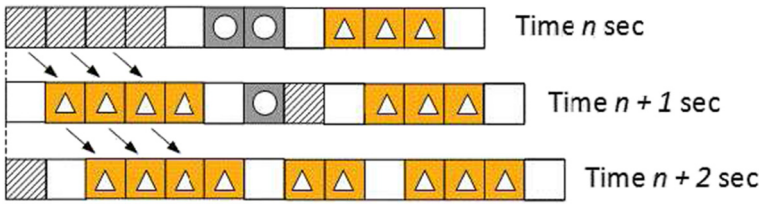

Fig. 2 Video organization. Trails (*hatched squares*) and video sequences (*squares with a triangle*) are created from the inflow and displacement of clients on the requested videos. Trails have less caching priority than sequences

on the left of the trail, the algorithm prevents the network access to obtain this content again with the main server. Henceforth, we will refer to each group of blocks laterally limited by two or more clients, as a *sequence*.

As shown in the scenario depicted in Fig. 2, several sequences can be formed along the service delivery time. As the demand increases, i.e., the number of clients to be served by the proxy becomes higher, the memory spaces used to allocate the video blocks for the clients (especially the new ones) need to be obtained from a recycling process of the available sequences.

In terms of efficiency, the main advantage linked to the design philosophy using sequences is illustrated in Fig. 3. It compares the logical video organization through sequences with the video organization through segments of equal size. In the figure, the two streams of video blocks, on the top and on the bottom, correspond to the same video clip. The top of the figure shows how the video clip is organized using segments of the same size where each segment contains four video blocks. The bottom part of the figure shows how the same video clip is organized utilizing sequences created from the watching position of the clients.

In the video organization through fixed size segments, any segment that is being accessed by a client cannot be removed from the memory. Consequently, for the video clip shown in Fig. 3, there are only two segments, totaling eight video blocks, that could be removed from the memory by a replacement algorithm. In contrast, in the video organization through sequences, only the video blocks that are currently being accessed by the clients cannot be removed from the memory by the replacement algorithm. As a result, all existing sequences in the video clip are eligible to be discarded, totaling 17 video blocks (among those present in Fig. 3).

Another example is assuming that the replacement algorithm has assigned a higher discard priority to the interval between the last two clients (on the left side of Fig. 3, for both strategies). The entire interval can be removed from the memory only if the video organization mechanism employs the sequences. If the video organization uses segments of the same size, the final part of the interval, which corresponds to the first two hatched blocks belonging to the segment $n + 2$, cannot be discarded despite its low probability in receiving an access in the future.

Thus, the sequence mechanism more precisely defines the areas of interest for the action of the replacement algorithm, enabling a better scalability of the proxy. For this reason, we chose to employ this scheme as a base for the implementation of our algorithm, described in a complementary manner in the next sub-subsection.

Caching logic

According to the video organization strategy described in the previous sub-subsection, the binding of priorities to memory spaces in our system considers that:

- Blocks that are currently being read by one or more clients cannot be discarded from memory.
- When free space is available, the memory employed to allocate the content coming from the main server should be obtained primarily from that portion of unused memory blocks.
- When free memory becomes scarce to allocate the content being streamed from the main server, the proxy tries to discard a trail of blocks left in the memory as a result of the service delivery made previously to a client. If there are no blocks in this condition, the algorithm selects a sequence for

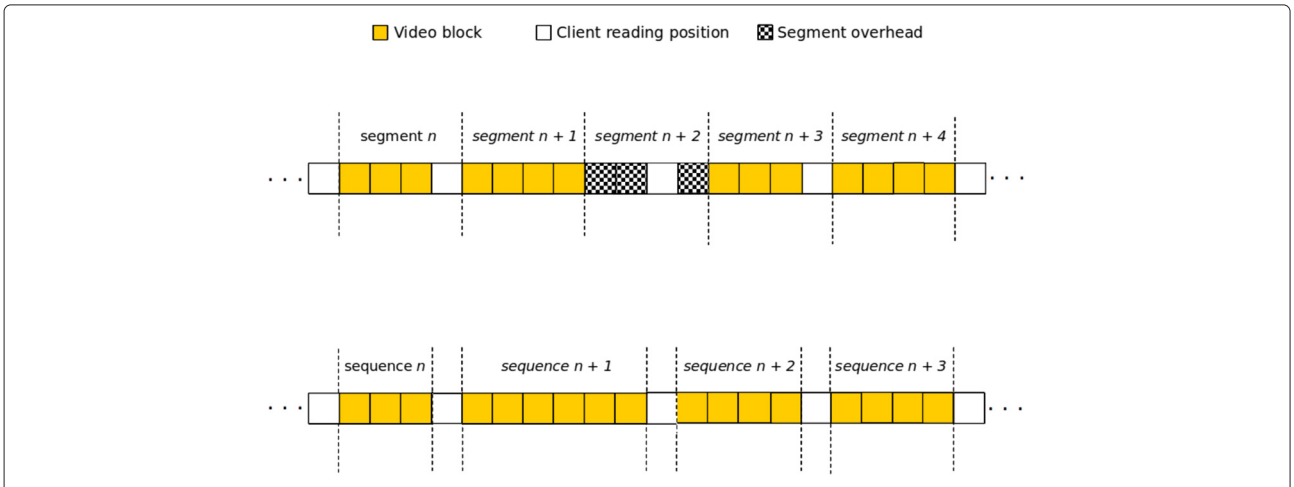

Fig. 3 Comparative: segments (fixed size) vs. sequences (variable size). Segments that are being accessed by clients cannot be discarded, while any sequence can be discarded to free space in the memory

discarding in order to free sufficient space to allocate the new content.

The victim sequence is chosen based on the caching priority of the available sequences, which is calculated incrementally along the sequences formation by combining the following factors:

- Start position ($SP_{s,v}$): corresponds to the current position of the first block (leftmost) belonging to the sth sequence of the video v, named S_v.
- Sequence size ($SZ_{s,v}$): number of blocks belonging to the S_v.
- Density of clients ($DC_{s,v}$): total number of clients present in a window of K blocks preceding S_v who potentially will perform access to this sequence.

This way, the caching priority for each sequence ($CP_{s,v}$) is defined by Eq. 1, where the higher the value for $CP_{s,v}$, the greater the probability of the sequence S_v is cached in the memory of the proxy.

$$CP_{s,v} = (DC_{s,v}/SZ_{s,v}) * [\, SP_{s,v}] \tag{1}$$

$$\text{where} \qquad DC_{s,v} = \sum_{i=1}^{TC_v} C_{i,v} \tag{2}$$

$$\text{suchthat} \quad C_{i,v} = \begin{cases} 1, & WB_{s,v} \leq P_{i,v} < SP_{s,v} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$\text{and} \qquad WB_{s,v} = SP_{s,v} - K \tag{4}$$

In Eq. 2, $C_{i,v}$ is a binary variable that states if the access position of the ith client of video v ($P_{i,v}$) is (1) or not (0) confined within the time window of the sequence S_v. TC_v informs the total number of clients that are watching the video v and $WB_{s,v}$, present in Eqs. 3 and 4, informs the current position of the left edge of the time window preceding S_v.

As shown in Eq. 1, our algorithm prioritizes the caching of the sequences with the highest density in the short term (indicated by the number of existing clients within the window of K blocks that precedes each sequence) since this choice benefits a larger number of clients. To uniformly apply this design decision, the sequences in formation at the video prefix that does not have previous K blocks yet can not be discarded as the clients density for these sequences can not be defined if the time window is not completely constituted.

In addition, Eq. 1 shows that the algorithm also favors the preservation of smaller sequences in the memory, since this choice allows for better scalability due to lower consumption of memory resources.

What is more, the variable $SP_{s,v}$, that appears in brackets in Eq. 1, is used as the tiebreaker and it is only multiplied by the result of the division when two or more sequences have the same value for $CP_{s,v}$. Accordingly, when a tie occurs, the sequences closer to the end of each video receive higher caching priorities, aiming to enable not only the existing clients within each window to perform access to these sequences but also all existing clients in the longer term.

Lastly, CARTE is able to support VCR operations such as "jump forward," "jump back," or "abort." When these events occur, the algorithm updates the density of clients of the time windows affected by the operations performed by clients. So for instance, when a client aborts the execution of a movie, the algorithm subtracts by one unit the density related to the window where the client was positioned at the time he or she decided to abort the video transmission.

However, we emphasize that, aiming to adopt the same strategy employed by the authors of the algorithms we selected (using the criteria described in the "Results and discussion" section) to carry out a comparative analysis with the CARTE, we chose to execute the greatest part of our simulations without the occurrence of VCR events. In this way, we configured an impartial comparative environment to assess the performance of our algorithm.

Experimental environment

As far as we know, in most publications in the field of developing proxies for VoD, the researchers have developed their own simulators (as was done in [18]), or they have adapted existing network simulators (as in [19]) to perform their experiments in order to avoid the cost and complexity inherent to the development of real environments for prototyping their systems.

In addition, these two alternatives have been most commonly used to assess the impact of new caching policies on the viewpoint of conventional metrics such as hit ratio, clients blocking ratio, network resources occupation, and startup delays for service providing, among others. However, in our view, both approaches are incomplete since they do not consider the analysis of architectural aspects of the proxy which are also potentially decisive for the performance and scalability of the system, such as the consumption of physical resources to run the application.

This way, we found inspiration to tackle the development of a new tool named PROxy SIMulator (SIMPRO) that is the core of the simulation flow used in this work. The following sub-subsection ("Synthesis and simulation flows") presents the flows for synthesis and simulation using our tool. The "Basics of the measurement process" sub-subsection details the general measurement process, exposing the main concepts and simulation alternatives used for data acquisition. Finally, the "Validation of the performance data" sub-subsection describes the process used to validate the data produced by the simulator.

Synthesis and simulation flows

Figure 4 illustrates the steps of the simulation flow managed by SIMPRO. The flow starts by the generation of the simulation input vector (SIV) which describes the workload for the proxy simulation. The basic element for constructing the SIV is a request to create a video session. Each request specifies the arrival time, the video identifier, and the session duration (used to simulate scenarios with early departure of the clients).

The SIV file is automatically generated from a set of parameters configured by the designer via *SIMPRO Conf* panel. In addition to the total simulation time, the parameters used for configuring the workload are as follows: the maximum number of requests (client arrivals), the quantity, the size and the bit rate of the videos available for access, and the distribution model for the dispersing of requests over the collection.

The requests distribution model is based on the use of Poisson [20] to define the interval between the arrivals of clients. The greater the value of the Poison coefficient (λ) is, the greater the average time between the arrivals of the requests to the proxy will be. In addition, the Zipf distribution [21] was used to specify the way each client will be associated to one video in the collection. The greater the Zipf coefficient (θ), known as "skew," the greater the concentration of the requests over the most popular videos of the archive, with an exponential decay. In other words, if the value of θ is closer to zero, the probability of access to the videos is more homogenous.

The next step in the simulation flow is to configure the proxy architecture that comprises of the following:

- *Caching Algorithm*: described and/or selected by the designer of the proxy.
- *Timing Manager*: designed to provide support for the synchronization of the proxy operations.
- *Network Interface Abstractor (NIA)*: used to store the SIV file and the video blocks that enter and exit the proxy.

During the simulation, at every second reported by the Timing Manager, the caching algorithm checks the NIA buffers for the presence of requests whose arrival time is equivalent to the current simulation time. For each like request, the algorithm creates a session handler and stores it in the buffer of active clients. Then, the algorithm reads each handler present in the clients buffer to discover the current reading position of the respective client. After that, it checks whether the requested content is available in the video memory. If so, the algorithm transfers the correspondent video blocks from this memory to the NIA buffers. Otherwise, it decides (based on its prioritization logic) if it should or should not transmit a video requisition to the main server (via Server Channel) to obtain this content.

Lastly, during the SIMPRO design, we took into consideration that, for the most popular videos, memory speed is a requirement as important as memory capacity, especially when the proxy is serving hundreds (or thousands)

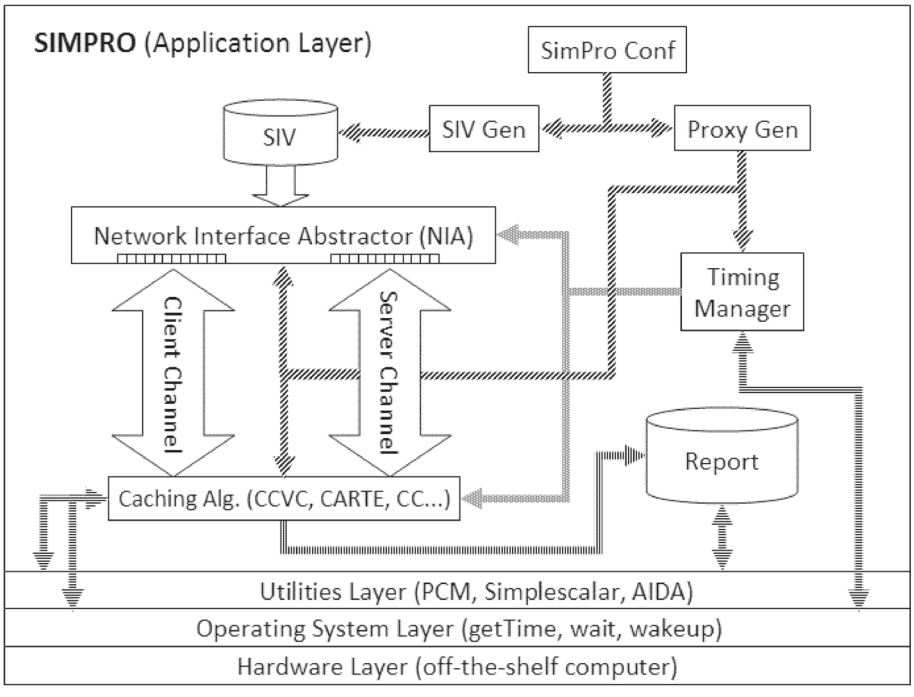

Fig. 4 SIMPRO. Flow for synthesis and simulation of a VoD proxy

of concurrent clients whose requests are concentrated on a few, more popular videos on the collection. In these conditions, even if there is plenty of space in the disk to fully allocate these videos, the disk bandwidth would not be able to sustain the required throughput to serve the active clients who are watching these videos [22, 23]. On the other hand, as demonstrated by [24], the high bandwidth currently supported by both the I/O buses and the network interfaces causes none of these components to conform a bottleneck to the target application, even in scenarios of high workloads.

Thus, we were concerned with designing a simulator to evaluate the proxy efficiency with focus on the performance of the processor and the RAM, since these devices are inexorably the main elements capable of providing, as well as limiting, the system scalability in face of the growing demand for the most popular videos on the system.

Basics of the measurement process

To clarify the performance evaluation of the proxy before the demand produced by a workload, we introduced the concept of *round of service* which consists in all the tasks undertaken by the proxy in order to serve each of its active clients with a predetermined amount of video[4].

Accordingly, the main operations that the system needs to perform during execution of a round of service are as follows:

1. Check for the presence of the requested content in memory and, when necessary, send a content request to the main server.
2. Receive the data requested to the main server and execute the replacement algorithm to release space in the memory to store the new content.
3. Admit new clients and close the sessions of the clients that ended watching the required content.
4. Update the caching priorities of the video objects.
5. Deliver the respective content to each client.

Besides, assuming that the proxy sends the content to its clients using a constant transmission rate equal to the (also constant) coding rate of the video, a new video block, containing 1 s of video, should be delivered to each active client at every second. Thus, in a scenario where the delivery of video blocks to the clients is always the last task performed in a round of service, we concluded that the proxy will have missed its deadline to serve part of its clients if a round of service takes a time longer than 1 s to finish. By using this condition as a reference, we defined the round of service as the unit of effort to evaluate the proxy performance and, consequently, all results produced by SIMPRO were primarily related to the performance of the system to execute one or more rounds of service.

Given this, as shown in Fig. 4, the last step of the simulation flow consists in generating the performance report of the proxy, which is done in accordance with the simulation mode selected by the user. Three simulation modes are currently supported by SIMPRO: (1) real time, (2) discrete time, and (3) functional.

In real-time mode, the simulator collects and temporarily stores in the RAM the execution time for each round of service. To enhance the data set produced by the real-time simulation with metrics like number of processor cycles and memory bandwidth consumed per round of service, the SIMPRO communicates with the physical level, using the Intel Performance Counter Monitor (IPCM) library [25]. IPCM utilizes dedicated hardware existing inside the processor to measure software performance without interfering on the execution time of the applications.

In discrete mode (slower simulation), SIMPRO interacts with Simplescalar [26] to produce additional information about the software–hardware interface, such as an instruction profile. This profile describes the instructions most frequently used by the software, allowing the designer to create architecture customizations to increase the VoD proxy performance.

Another advantage provided by integration with Simplescalar is the capability to evaluate the application performance using different architectures. This is especially important considering that the IPCM is restricted for use with Intel processors. Thus, we can alternatively use discrete mode to evaluate the same metrics measured by IPCM when Intel processors are not the target for the evaluation or when the physical target processor is not available to execute the simulation in real-time mode.

Lastly, in the functional mode (fast simulation), our simulator produces values for hit ratio[5], number of block substitutions, and network bandwidth demanded in each round of service, as well as other more detailed information (if requested by user), such as system debugging data. In this mode, the proxy performs all its operations to preserve the coherent state of the system along the execution of the rounds of service, except the data I/O to move video block to and from the RAM. As a result, the simulation time becomes significantly faster than the one obtained through real-time and discrete modes, allowing the designer to get a quick feedback in terms of the metrics most widely used to perform both behavioral and efficiency analysis of a VoD proxy.

Validation of the performance data

A complete description of the CCVC algorithm (briefly introduced in the "Background" section) followed by performance data presented in [27] was used with the purpose of reproducing and integrating this algorithm to SIMPRO to generate comparative data aiming to validate the simulator. The simulation parameters used by the

Table 1 Workload parameters for functional validation. Data originally used in [27] for validation of the CCVC algorithm

Workload parameter	Value
Zipf coefficient - θ (skew)	0.271
Poisson rate - λ (sec)	3
Number of videos	100
Cache size (percentage of the collection size)	10
Simulation duration (min)	60
Video transmission rate (Mbps)	1
Video length (min)	60 (= 450 MB)

CCVC authors are described in Table 1 and the original performance results (about the network consumption in the main server-to-proxy channel) used for comparisons are presented in Fig. 5a (see the curve labeled as "Reference"). In the same figure, for the same loading conditions, we plotted the results produced by functional simulation performed by our simulator.

As shown in Fig. 5a, the results obtained through SIMPRO simulations[6] are almost identical to the original results with a maximum difference of 2 % (for time = 58 min[7]). This permits us to expect that the results produced by SIMPRO in functional mode are valid, although we are not able to validate the entire set of metrics measured by our simulator due to the unavailability of measurements performed with the same metrics by the CCVC authors.

In our analysis, we have used the functional mode to measure the hit ratio produced for all algorithms. The main reason for this choice is that this metric is less restrictive than channel consumption since the bandwidth associated to the proxy-main server link may be very limited in some operating scenarios.

Regarding the validation of the real-time and discrete-time modes of our simulator, it was necessary to create a set of benchmark results by collecting data performance of the VoD proxy during its execution in a more realistic context which we refer to as "networked implementation".

For this implementation, we used two Dell PowerEdge M620 computers, both equipped with two Intel Xeon E5-2600 processors, 16 GB of DDR3 1066 MHz RAM and Broadcom 57810S-k dual-port 10 gigabit NIC, capable of performing TCP/IP operations directly in hardware (TOE technology). Both computers run SUSE Linux Enterprise Server (3.0 kernel), and they were connected to each other using a 10 gigabit optic fiber channel. All network traffic between these computers was implemented directly in the C source code using the sockets programming library.

To reproduce a typical VoD operation scenario, we configured one of the computers to run the VoD proxy using the CCVC algorithm and the workload presented in Table 1. The other computer was configured to perform simultaneously the behavior of the clients and the main server of VoD system. In this experiment, the video blocks received from the proxy were not decoded after the receipt.

The validation was performed by measuring and comparing the service time (time to perform one round of service) resulting from the execution of the networked implementation and the service time resulting from the execution using SIMPRO in both real-time and discrete-time modes. We used the IPCM software to measure the processing time produced by the networked implementation, taking into account all delays caused by the RAM. To obtain the service time by means of the discrete mode of the SIMPRO, we measured the number of cycles by running the proxy on a PISA architecture, available on the Simplescalar toolset, which according to [28] has several similarities with the physical processor we used in these experiments. After that, the number of cycles was multiplied by the clock period of the target processor.

As shown in Fig. 5b, both the discrete- and real-time modes produce results very close to those produced by the networked implementation. The error produced by the real-time simulation was not larger than 5.8 % (for time = 26 min), while the error produced by discrete mode was not higher than 6.2 % (for time = 59 min). These errors are acceptable according to similar results presented in [29–31] for experiments of an equivalent nature.

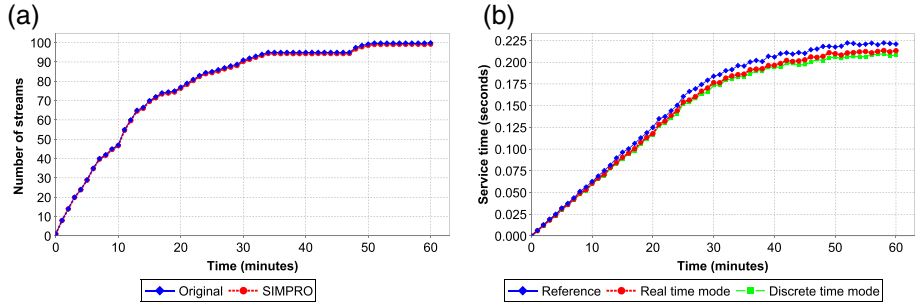

Fig. 5 Validation of the SIMPRO. Comparative analysis of the results to verify the accuracy of the data generated by (**a**) functional mode and by (**b**) the discrete- and real-time modes

Considering that we configured our simulator to not consider the TCP/IP processing cost, based on the capabilities of the TCP Offload Engine (TOE [32]) network interfaces, we attributed these differences to the execution of some TPC/IP operations on the processor, since, unfortunately, the TOE technology available on the computers used in this experiment does not support the full layer set of the TPC/IP protocol.

Results and discussion

This section presents and discusses results of the performance comparison of the three video caching algorithms previously presented:

- CCVC and CC (both introduced in the "Background" section), since they are pioneers in the proposition of the *Look Ahead* caching paradigm.
- Our algorithm, named CARTE, described in the "Proposed algorithm" subsection of the "Methods" section.

As a strategy to obtain each result, 30 simulations were executed obtaining confidence of at least 95 % to an interval of confidence less than 1 % of measurement value.

The following subsections are organized as follows: the "Workload and operating parameters for the evaluation of the proxy" subsection describes the workload and operating parameters to run the simulations. The subsection entitled "Evaluation of the hit ratio and service time for the execution of the caching algorithms" analyzes the hit ratio and service time produced by the algorithms and the "Evaluation of the amount of computational resources used by each algorithm" subsection shows the evaluation of the computational resources consumed by each of them. The subsection named "Impacts produced by the early departure of clients" analyzes the performance of the algorithms when executed in scenarios with early departure of clients with the aim of demonstrating that our algorithm also shows better performance under these load conditions. The "Analysis of the relation between the size of the time window and the workload" subsection presents a study of the impacts on the hit ratio caused by the variation of the main design parameter of CARTE: the size of the time window used to define the space for the count of the clients who are in front of each video sequence. Finally, the "Procedures to configure CARTE" subsection shows the strategy currently being used to configure our algorithm adequately for each scenario.

Workload and operating parameters for the evaluation of the proxy

The results presented in this subsection were all obtained from simulations performed on a general purpose server organized with two Intel Xeon E5530 (2.4 GHz), 16 GB of DDR3 1066 MHz RAM and SUSE Linux Enterprise Server (3.0 kernel).

The topology assumed for the experiments considers scenarios where all access requests for the content provided by the VoD system are routed through the proxy. This way, using the current position of the active clients, each caching algorithm determines (based on its prioritization logic) which video chunks must be preserved in the memory (if the chunks are already allocated in memory) or requested to the main server. Moreover, the requests for video chunks to the main server are done assuming an existence of a limited bandwidth in the link that connects the proxy to this server, as occurs in real scenarios.

Table 2 shows the configurations used for building the workloads and other operating conditions employed in our simulations. The set of acronyms presented in the table were used to substitute the full names of the working parameters along this section.

The experimental methodology consisted of selecting one-by-one, each of the working parameters for variation and evaluation, and fixing all other parameters using a default value. This way, we expected to clearly observe the influences caused by each parameter on the efficiency of the system. For this purpose, the third column on Table 2 shows the standard value to be used when the concerned parameter is not under variation. Otherwise, the fourth and fifth columns, respectively, present the range and the increment step to perform the parameter variation.

The interval for variation is obtained by proximity in relation to standard value. The motivations for the use of the standard values presented in Table 2 are described below:

- *TR*: the configuration adopted corresponds to the minimum bandwidth necessary to stream movies with HD quality [33].
- *NB*: in [8], it was demonstrated that the average requisite for NB, for similar workload conditions, is not superior to 700 streams. This value was used as standard configuration, except during the variation of the parameter IA, where the standard value of NB was reduced to 200 streams to prevent all algorithms from obtaining a high hit ratio (in response to the increase in IA) hindering in this way the observation of impacts caused by modification of this parameter.
- *MS*: according to the explanations made in the sub-subsection entitled "Synthesis and simulation flows" ("Methods" section), we set the standard value for MS parameter to the maximum quantity of RAM available on the computer utilized to perform the experiments, allowing the execution of simulations with the highest possible workloads.
- *VL*: the used settings followed the estimates presented in [34–39], for the average length of movies made available by the VoD providers.

Table 2 Working parameters

Acronym	Parameter name	Standard value	Range	Step
TR	Video transmission rate (Mbps)	5	–	–
NB	Max. proxy-server net. bandwidth (streams)	200; 700	200–700	50
MS	Memory size (GB)	14	4–14	1
VL	Video length (min)	90 (=3.4 GB)	90–240 (=3.4–9.15 GB)	30
NV	Number of videos	100	50–150	10
ZC	Zipf coefficient - θ (skew)	0.271	0–1	0.25
IA	Average interval between arrivals - λ (sec)	3	1–10	1
SD	Simulation duration (min)	VL + 1500 rounds of service	–	–
TW	Time window size (sec)	–	1–300	1
SS	Segment size (seconds of video)	1 (≈0.63 MB)	–	–

The third column shows the standard value to be used when the parameter considered is not under variation. Otherwise, the fourth and fifth columns present the range and the increment step, respectively, to perform the parameter variation

- *NV*: taking into account the information provided in the sub-subsection entitled "Synthesis and simulation flows" ("Methods" section), the standard configuration was inspired by the quantity of new (and highly popular) videos introduced daily/weekly by the VoD providers[7] [34, 40, 41].
- *ZC*: the standard value for the this parameter has also been used for modeling the video popularity skew of video rental scenarios in [34, 42–46].
- *IA*: according to [47] and [16], the configuration used corresponds to a typical scenario of high workload for VoD proxies for similar video lengths.
- *SD*: we expected during the extra interval of 1500 rounds of services both the entry and exit of clients in the system. So, to produce each basic result, we measured the average performance during this time interval since, in this way, each presented result represents the performance obtained for a full operation of the VoD proxy.
- *TW*: for each scenario, we evaluated the best configuration for our algorithm by varying the size of the time window in a range of 1 to 300 s. This way, along comparisons with the other algorithms, we informed above each result presented for CARTE the configuration of TW that produced the best performance for such scenario.
- *SS*: in all of the experiments, the size of the video segments used for CC algorithm was equivalent to one video block of 1 s of video. This choice was made in order to prevent the CC algorithm from experiencing a potential drop in its hit ratio due to the waste of memory that may be caused by the use of larger segment sizes, as explained in the "Video organization" sub-subsection of the "Methods" section.

Following the same strategy employed by the authors of the algorithms we selected to carry out a comparative analysis with the CARTE, we chose to use experimental scenarios (except those mentioned in the "Impacts produced by the early departure of clients" subsection) where the clients require the video from the beginning to the end. Therefore, we intended to configure a fair comparative environment to assess the performance of our algorithm.

Evaluation of the hit ratio and service time for the execution of the caching algorithms

According to the workload and operating conditions described in the previous subsection entitled "Workload and operating parameters for the evaluation of the proxy", the graphics in Fig. 6a–j describe the impacts produced by the variation of the working parameters on the average hit ratio (graphics a, c, e, g, and i) and service time (graphics b, d, f, h, and j) produced by the analyzed algorithms.

A first and general conclusion about the data presented is that, at the cost of a slightly higher service time (due to the more complex replacement mechanism), the hit ratio produced by CARTE is higher than that provided by CCVC and CC.

As demonstrated by Fig. 6a, c, i, and k, the increase in NB, MS, ZC, and IA also produced the increase on the hit ratio for all algorithms. Respectively, this occurs because of the following:

- When taking into consideration the limitations imposed by the storage capacity of the memory, the increase on NB enables the proxy to carry out a greater volume of substitutions on the cache, thereby also allowing a greater number of clients to have their demands met.
- The higher the value for MS, the greater chances of the stored content to remain in the memory to be accessed by other clients.
- Increasing ZC causes the demand of the proxy to be concentrated in a few videos of the collection. As a

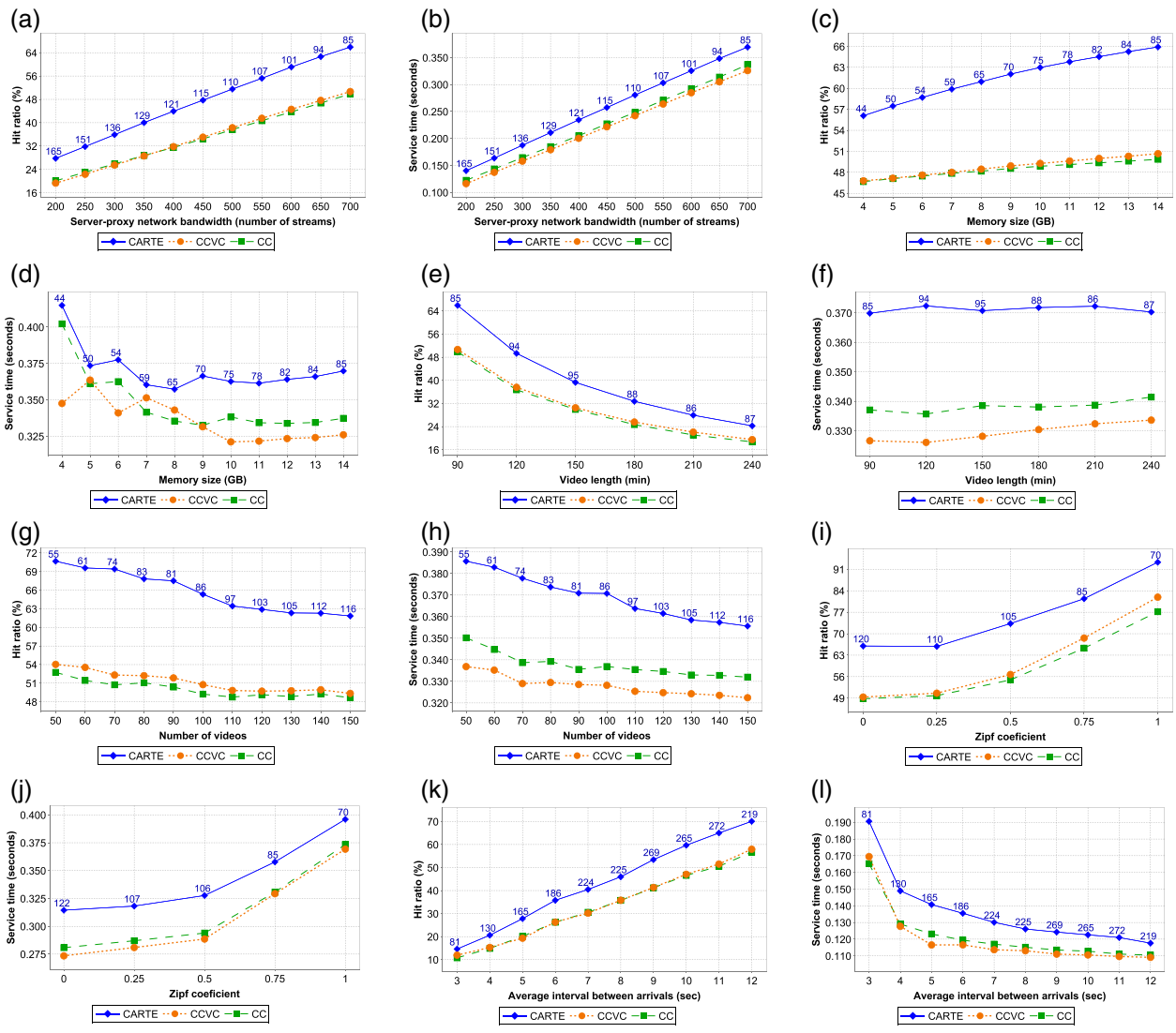

Fig. 6 a–l Performance of caching algorithms. Results for hit ratio and service time

consequence, the memory requirements to support the active clients are reduced and a higher number of blocks belonging to the requested videos will remain in the memory to be accessed again by newer clients.

- A larger value for IA, for a fixed video length, makes the number of active clients of the proxy to decrease. Accordingly, both memory and network requirements to meet the demand are lowered causing the system performance to increase.

Still with regard to Fig. 6a, c, and i, it is important to highlight two aspects taking into account the performance of our algorithm. Firstly, the greater the availability of resources (memory capacity and network bandwidth) is in relation to the demand, the higher the hit ratio provided by CARTE in comparison to that one provided by the other two algorithms is. Secondly, CARTE achieves a comparatively higher hit ratio even in scenarios where the popularity of the videos leans towards equality (i.e., when ZC assumes its lowest values). From this, we concluded that our approach is not only in fact more scalable but also is capable of leveraging the performance of the proxy even in the worst-case scenarios when the video access probability cannot be easily predicted, such as it occurs, for example, at certain periods for the most popular items of a collection [48].

In addition, CARTE presents a higher performance when ZC assumes a value even greater than the ones shown in Fig. 6i. Considering, for example, a scenario where the standard configurations are used except for the ZC value which is configured to be equal to 5 (higher skew), making MS equal to 2 GB produces a difference

between the hit ratio of CARTE and of CCVS at 8.1 % while the difference for CC is 11.8 %. The higher performance provided by our algorithm results both from allocating the smaller sequences in the memory, which results in a better use of the storage resources, as well as considering at a local level the differences that exist between the densities of clients formed along the arrivals of the clients. The latter factor contrasts with the behavior of the CCVC and the CC, which only look at clients as a whole who watch a particular video to decide which portions of the contents should be kept in the memory.

Contrary to what occurs in graphics a, c, i, and k of Fig. 6, graphics e and g of this figure demonstrate that the increase applied to the working parameters produce the decrease of the hit ratio for all algorithms. This opposite effect occurs because of the following:

- For each increase on VL, while the new clients request the access to the prefix of each video, the older clients continue moving towards the more distant end of the video, taking longer to conclude their session. This causes the number of active clients to increase and to become dispersed over a larger video space, resulting in higher memory and network requirements, dropping the hit ratio.
- Increasing the number of videos causes the clients to be spread over a larger collection. Consequently, since ZC remains constant along the experiment, these clients are directed to portions of the collection that have not been accessed yet by other clients or that have been accessed for a long enough period of time to cause it to be removed from the memory.

By analyzing the information available on the right side of Fig. 6, it is possible to see that in graphics b, h, and j the service time for each algorithm, in general, follows the growth produced on hit ratio. This occurs because each algorithm needs to employ an effort to transfer video blocks from the local memory to the network interface in proportion with the number of clients to be served. However, in the scenarios represented by the graphics d, f, and l of Fig. 6, the service time does not follow the growth of the hit ratio. This occurs because of the following:

- The larger service time obtained for the smaller memory sizes is resulting of a greater processing to continuously change the content available in the cache, aiming to minimally store the video blocks that are a step forward in the execution line of the clients.
- If VL is small, the number of active clients is also small. However, when VL is small, the availability of resources compared to the number of active clients is greater and the hit ratio is higher. Following an inverse reasoning, as VL increases, the hit ratio decreases, but the number of active clients increases. Thus, the number of clients actually served by the proxy tends to remain roughly constant along the variation of VL and, as a result, the service time also remains nearly stable during this period.
- As IA increases and VL remains constant in the experiment, the number of active clients is reduced, causing the drop of the service time.

The average differences in the number of hits and simulation time between CARTE and CCVC for all simulation scenarios were 12.9 and 11.3 %, respectively. For the same metrics, the differences obtained when the working parameters assumed their standard values were of 14.6 and 11.6 %, respectively. Lastly, the maximum difference in the number of hits between these algorithms, obtained during the variation of the parameter NV (Fig. 6g) when NV was set to the value of 70, was 17.3 %, for a corresponding difference in the service time of 12.8 %.

There are two reasons for the better hit ratio obtained by the CARTE. First, as mentioned before, CCVC employs a Least Frequently Used (LFU) algorithm in the beginning stage of the replacement process to define from which video the sequences will be discarded. Therefore, unlike the policy executed by the CARTE, which assumes that the sequence to be replaced can come from any cached video, the strategy used by CCVC reduces the space of alternatives for the action of the replacement algorithm, since the selection of a victim sequence must respect the scale of popularity (access frequencies) of the videos.

Secondly, once the video of smaller popularity has been chosen, CCVC selects the sequence closer to the end of the video for disposal since in this way (according to the view of the CCVC authors) the proxy needs to employ its resources for a shorter time to recover back the removed content with the main server to serve the clients who are moving toward the video sequence removed from memory. However, since the sequences move jointly with the movement of the clients, even the sequences that have a greater quantity of previous clients in the short-term tend to be disposed when they begin to get close to the end of the video. As a result, the efficiency of the CCVC algorithm tends to decrease due to this priority inversion during the final stage of video transmission to the clients.

On the other hand, the performance comparison with the CC proxy has shown higher gains. The average difference in the number of hits was 13.8 % for all simulation scenarios and 9 % for the service time. The differences obtained for the same metrics when the working parameters assumed their standard values were 16.1 and 11.4 %, respectively. The maximum difference in the number of hits, also obtained during the variation of the parameter

NV when this parameter was set to the value of 70, was 18.9 %, for a corresponding difference in the service time of 10.4 %.

As mentioned in the "Background" section, the origin of the lower hit ratio related to the CC system relies on the tendency of excessive prioritization of segments positioned at the end of the video. This occurs because this algorithm calculates the total number of active clients who have not yet requested access to a given video segment as a way of estimating the number of hits which will be performed in the future for this segment. Consequently, the algorithm gives preference to storage of the long-term demands, instead of favoring the short-term ones, causing a drop in its performance when the momentary load conditions points to the storage of the initial or intermediate portions of the videos.

Evaluation of the amount of computational resources used by each algorithm

Figure 7 depicts, for the same system and load configurations used in the construction of Fig. 6, how the physical resources of the target architecture are used. This set of data was obtained through simulations performed in real-time mode supported by SIMPRO.

Using the same operating specifications, we collected an instruction profile which is also presented in this subsection. To obtain this supplementary data, we ran simulations using the discrete mode of our simulator. For this purpose, we instantiate a PISA architecture following the same strategy we used to perform the SIMPRO validation (described in the "Validation of the performance data" sub-subsection).

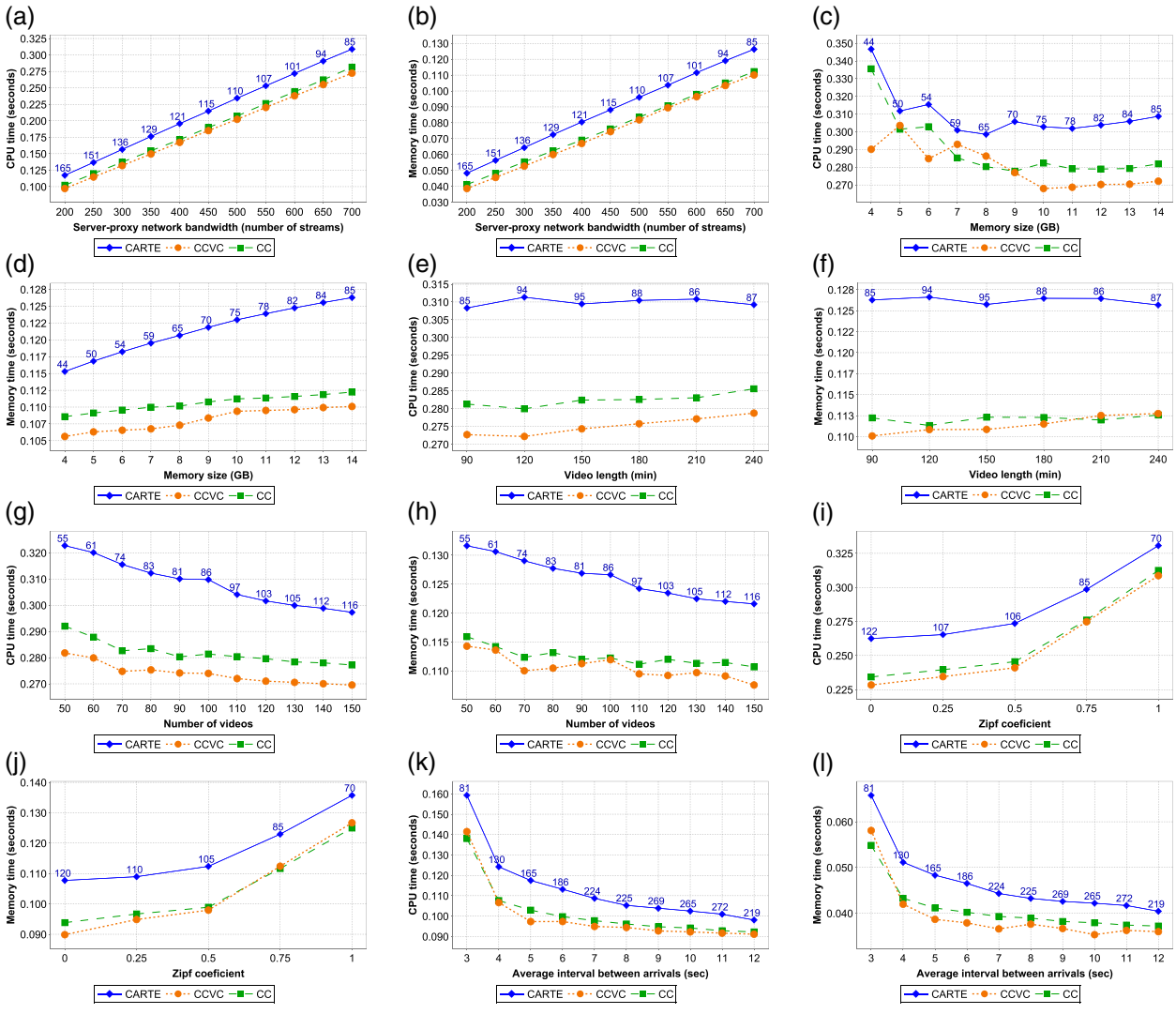

Fig. 7 a–l Resource usage. Results for CPU and memory time

In Fig. 7, the graphics on the left (a, c, e, g, and i) describe the time consumed exclusively by the execution of instructions on the processor without delays produced by the memory system. The graphics on the right (b, d, f, h, and j) describe the time required to perform data transfers to and from the memory. This memory time considers the full delay necessary to perform all memory transfers that includes, besides the I/O flow of video blocks, the traffic resulting from the access to the program variables and the data structures used to support the proxy operations.

In most graphics available in Fig. 7, the CPU and the memory follow, although not in the same proportion, similar behavioral trends. This occurs because the effort performed by the proxy is dominantly determined by the number of clients served by the system. However, under certain circumstances such as that represented by Fig. 7c, d, the behaviors of the times of the CPU and the memory may differ.

This difference occurs because as the memory size decreases, the algorithms execute a greater processing (resulting in a larger CPU time) to continuously promote the exchange of the video blocks available in the memory aiming to allocate the next content that will be accessed by the clients. However, as the memory is reduced, the hit ratio also decreases, resulting in a lower volume of clients served by the proxy and, consequently, a shorter memory time to transfer the required amount of video blocks from the memory to the network.

Furthermore, as shown on the full set of results available on Fig. 7, the processor consumes on average about two to three times the time required by the memory to execute the application. The difference between the time consumed by these components demonstrates that the processing core tends to be the main bottleneck for the service provision when the demand increases relative to the amount of resources available.

The analysis of the data produced by the simulation in the discrete mode has revealed that, on average, the arithmetic instructions account for the majority of the executed instructions by 63.2 %, whereas memory access and control flow account for 28.32 and 8.38 %, respectively. These results, in conjunction with those produced by the simulation in real-time mode, suggest that the development of a dedicated hardware to support the basic proxy operations tends to contribute to a significant increase in the efficiency of this application while reducing the costs for implementing a video on demand system.

Taking into account the comparative analysis of the resources consumed by each algorithm, the results demonstrated that our strategy demanded on average 11.4 % more processor time than CCVC and 8.7 % more than CC, for a difference in memory time of 12 and 9.7 %, respectively. When the working parameters were set to their standard values, the processor time spent by our algorithm was 11.8 % higher than the one consumed by CCVC and 8.9 % higher than the one spent by CC, also consuming 12.8 and 11.1 % respectively more time to perform the necessary memory accesses.

Moreover, the differences found in terms of processor time in relation to CCVC and CC, considering the scenarios where CARTE obtained the largest differences in hit ratio to these algorithms, were 12.6 and 10.1 %, respectively. Considering these scenarios, the difference found in memory time for comparisons between CARTE and CC was 12.4 %. Comparing CARTE and CCVC, the difference found for this metric was 14.7 %. The origin of these differences is based on three main aspects:

Firstly, the processing time tends to maintain an approximate proportion with the number of clients served by the system. Consequently, the gaps between the hit ratios produced by the algorithms contribute predominantly to the achievement of the differences observed between the consumption of resources.

Secondly, the processor time produced by the CC algorithm is also a consequence of the higher management cost attached to the video organization into segments of equal size. While this strategy enables the CC algorithm to obtain its highest hit ratio (according to the description made in the subsection named "Workload and operating parameters for the evaluation of the proxy"), it also contributes to an increase in the processing time of this algorithm. This occurs because, unlike the CARTE that binds a unique caching priority to multiple video blocks video encompassed into each sequence, the CC system assigns one priority to each video block allocated in the memory, thus requiring more processor time for this work. The mechanism for binding priorities used by CC has not produced a sufficient increase in the use of resources on the point of exceeding the consumption produced by CARTE (which mainly results from its higher hit ratio), but it does cause the CC to utilize more resources than CCVC, although the hit ratio produced by the CC is lower than that produced by CCVC in most scenarios analyzed.

Thirdly, the criteria used by both the CCVC and the CC to determine the caching priority for each chunk of video results in a smaller number of content substitutions on the cache, thus producing a lower consumption of resources to execute this task. In regard to the CCVC, this occurs because the algorithm concentrates the content replacements over the videos with fewest active clients and prioritizes the removal of the final portions of these videos. In this manner, only one significant change in the popularity of the videos can modify the caching priorities of the sequences and produce a higher number of content substitutions. However, such a change in the popularity of the videos rarely occurs in intervals shorter than a service day [15].

Regarding the behavior of the CC algorithm, although there is a tendency for the segments closer to the end of each video to receive a higher caching priority in relation to the segments closer to the beginning, the algorithm tends to enable a greater competition for the spaces available in cache when compared to CCVC. This is because at certain moments the load conditions may suggest that some segments near the beginning of a video of greater popularity have higher precedence over the segments closer to the end of a less popular video. In this context, the CC algorithm creates conditions for the execution of a larger volume of content replacements, consuming to this end a corresponding amount of resources.

In contrast, our algorithm was designed to allow any video chunk belonging to the collection (regardless of the position occupied within each respective video) to receive a higher caching priority, basing its caching decisions exclusively on the current load conditions inside each time window. As a result, the algorithm tends to react more frequently to the variations that occur in the distribution of the clients along the execution, thereby performing a higher volume of content updates on the cache with proportional impacts on the resources consumption.

Impacts produced by the early departure of clients

To model the behavior produced by the proactive session closing by VoD clients, we used the abandonment curve shown in Fig. 8, which was created from the data produced during the accesses to a commercial VoD proxy [34].

To analyze the impacts of the early exit of clients, we have reassessed the performance of the caching algorithms in the scenarios of Fig. 6c, f, as the memory size and the length of the videos are the parameters most closely related with the abandonment phenomenon [34].

Under the effect of session abandonment, the average and maximum differences between the hit ratio of CARTE and of CCVC were 8.6 and 11.8 %, respectively. These results represent a decrease of 2.2 and 3.4 %, respectively, in comparison with the scenarios without abandonment.

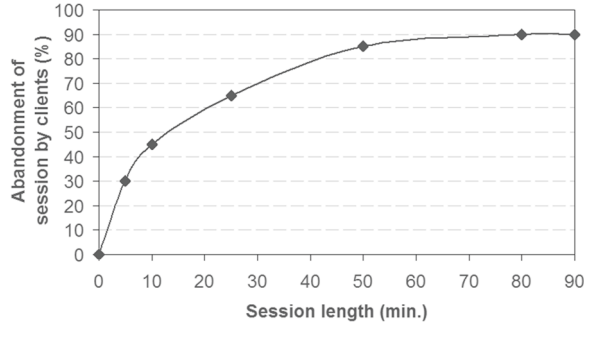

Fig. 8 Abandonment curve of VoD clients. Percentage of cumulative closing of sessions to each moment of video exhibition

The difference obtained between these two systems when default configurations were utilized was 11.8 %. This result represents a decrease of 3.4 % in relation to the difference obtained without abandonment.

This data shows that the CCVC can improve its performance compared to CARTE by discarding the sequences close to the end of the video[8]. However, CARTE produces a greater hit ratio because under the occurrence of abandonment CCVC does not discard sequences that belong to the videos of greater popularity where the session abandonment also happens. Consequently, when the clients abort a popular video, the sequences at the end of the video, kept in the memory by CCVC, need to wait more time until the next clients (who did not abandon the video) carry out access to them. As shown in Fig. 9a, this effect caused by the early departure of clients over the CCVC performance is even greater when the length of the video increases in relation to the session length of the clients. In contrast, CARTE only keeps the sequences in the memory when there are enough clients within their windows to justify their allocation. In this way, the algorithm tends to favor the allocation of sequences closer to the beginning of each video that has greater concentration of previous clients due to the gradual reduction of the number of active clients as they move forward to the end of the video.

In regard to the comparison with CC, the average and maximum differences between the hit ratios were 15 and 20.7 %, respectively. These results represent an increase of 3.6 and 4.7 %, respectively, in comparison with the scenarios without abandonment. The difference obtained between these two systems when default configurations were utilized was 19.9 %. This result represents an increase of 3.9 % in relation to the difference obtained without abandonment.

This happens because CC tends to prioritize the caching of the final portions of each video which are less accessed due to the early exit of clients. This occurs because the higher concentration of clients at the beginning of the video causes the increase of the caching priorities of the video blocks at the end of the video. So, as the video length increases in relation to the session length of clients, the numbers of blocks with greater caching priority after the exit points of the clients also increases. Consequently, these blocks dominate the spaces in cache resulting in an inefficiency of the CC algorithm.

In contrast, the windowing mechanism employed by CARTE causes the greater density of clients existing at the beginning of each video to only favor the sequences that are also located at the beginning of these videos. Consequently, the sequences situated near the suffix (end of the video) tend to not be kept in the memory since the densities in their windows are small compared to the windows of the sequences which are located at the prefix. In this

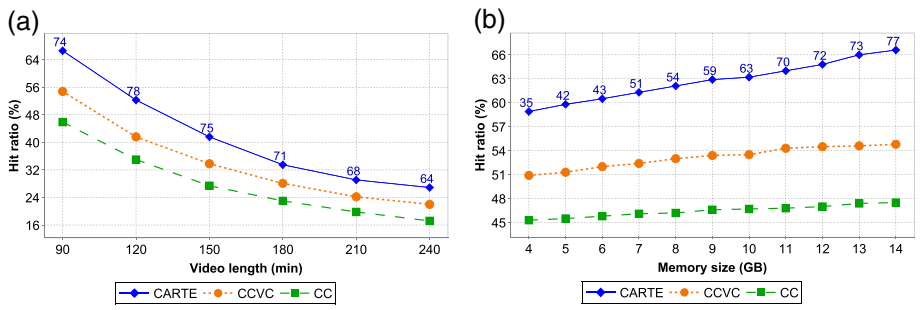

Fig. 9 a, **b** Performance under session abandonment. Hit ratios for variation of the video length (**a**) and memory size (**b**)

way, the accuracy of the algorithm to distinguish the different densities along each video tends to increase as the size of the time window becomes smaller. For this reason, the use of smaller time windows in the scenarios with abandoning caused CARTE to produce the best results.

The results in Fig. 9b show that, under the occurrence of abandonment, CARTE scales its performance better than the two reference algorithms when the size of the cache increases. This demonstrates that the caching logic of our algorithm is capable of exploring additional memory with greater efficiency by caching the sequences with higher concentrations of previous clients along the videos.

Finally, the service time produced by CARTE in the scenarios with session abandonment showed an average reduction of 7.8 % in relation to the analogues scenarios where the early departure of clients did not happen. This reduction came from a smaller number of simultaneous clients served by the proxy due to the abandonment phenomenon. Similarly, the reduction of the number of clients showed to be a major factor in increasing the average hit ratios produced by CARTE and CCVC by 1.5 and 3.7 %, respectively, when compared to the results obtained to the same scenarios where abandonment did not happen. CC did not follow this growth showing a decrease in its average efficiency by 2.1 %.

Analysis of the relation between the size of the time window and the workload

As suggested in the "Evaluation of the amount of computational resources used by each algorithm" subsection, the configuration adopted for the TW parameter of the CARTE is capable to create direct impacts on the hit ratio, as well as on the number of substitutions performed by this algorithm. As previously demonstrated, these two factors create repercussions on the consumption of resources of the underlying architecture. Consequently, the key aspect to reach the maximum efficiency of our algorithm is to identify the best fit for the TW.

For this reason, this subsection presents a study of the correlation between the TW and the workload for the proxy, aiming to demonstrate how the operating conditions could influence the choice taken by the VoD designer to permit the achievement of the best system response. To this end, we demonstrated the effects produced by the variation of the size of TW, under different scenarios, on the main metric of efficiency for a caching algorithm, the hit ratio. With this, we hope to build guidelines to perform the adjustment of our algorithm in order to extract its maximum performance for every case.

That said, Fig. 10a presents the existing correlation between IA and TW. The variation on the IA causes a change in the concentration (density) of clients per video. When IA assumes a lower value, the concentration of clients becomes higher and TW needs to be configured with a small value to better define the caching priority for each sequence. On the other hand, in scenarios where clients are more dispersed (high IA), it is necessary to use a larger TW to deal with the memory content more efficiently.

This occurs because when clients are more concentrated, the use of a large time window tends to encompass not only a great number of clients but also a high quantity of sequences within each window, making the same clients account for different sequences simultaneously. Therefore, the sequences closer to the end of the video tend to have more previous clients encompassed in TW space and consequently receive higher caching priorities. In contrast, the sequences close to the beginning of each video receive lower caching priorities, although these sequences might have a greater demand associated to them in the short term. When this occurs, the behavior of the CARTE algorithm tends to be closer to that of the CC algorithm.

Furthermore, when IA is equal or near to 1 s, producing the highest numbers of concurrent clients used in our simulations, even the best configuration for TW leads to the observation of significantly smaller gains on the hit ratio when compared to the gains obtained in the scenarios where IA assumes a higher value. This happens because, under these circumstances, both the size of the sequences

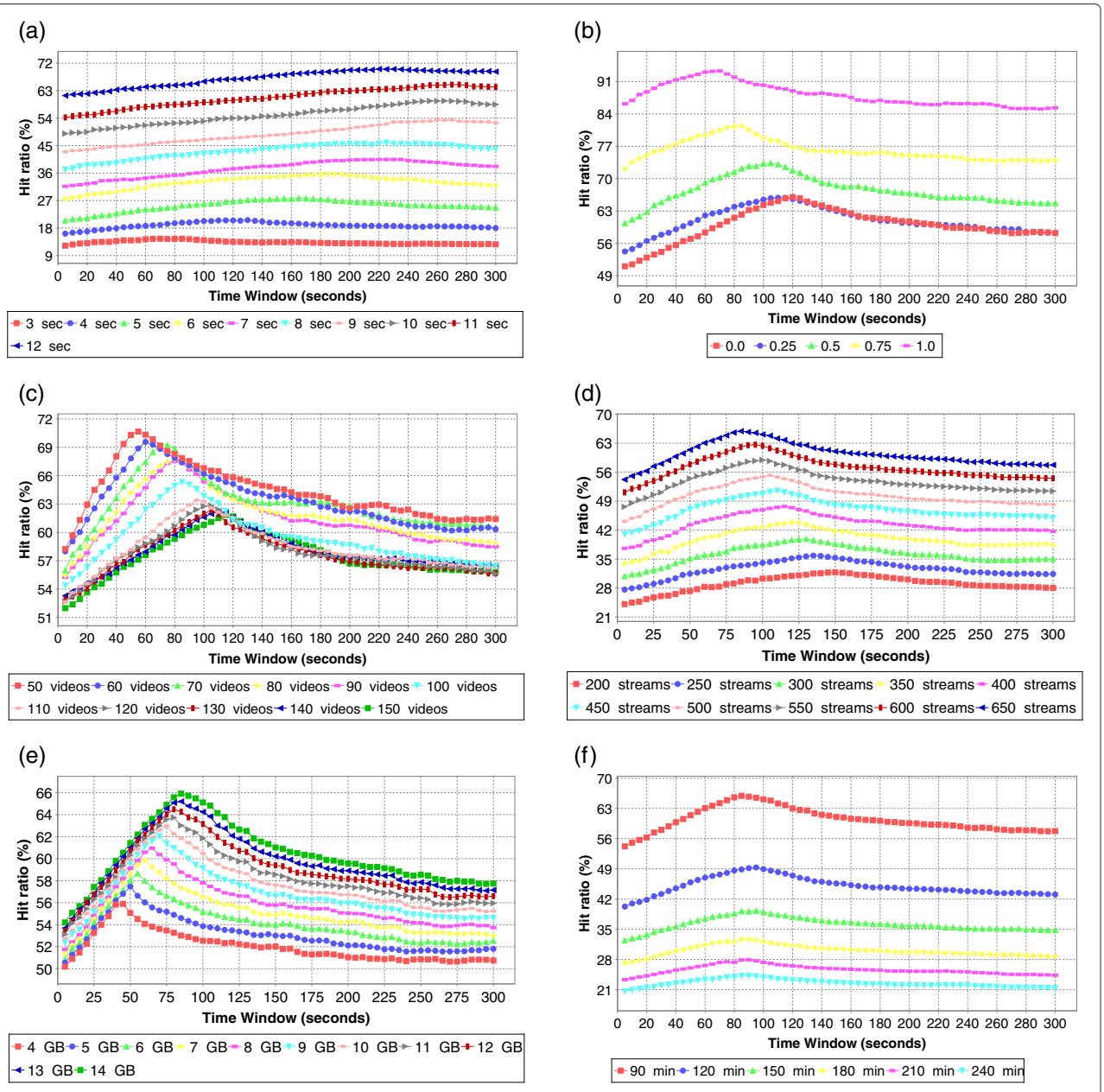

Fig. 10 Hit ratio produced by the variation of the size of the time window. In each graphic, TW varies in conjunction with one of the load parameters: IA (**a**), ZC (**b**), NV (**c**), NB (**d**), MS (**e**), and VL (**f**)

and the density of the clients within each time window of the victimized videos become similar, causing the benefit of maintaining one or other sequence in the memory to be almost the same.

Figure 10b shows the influence caused by the variation of the ZC on the choice of the configuration for TW. Considering that the NV and IA were kept constant in this experiment, each increase produced in ZC caused the transfer of a significant volume of clients from less popular videos to the most popular videos. For this reason, a few clients remained in the most popular videos, resulting in a smaller number of sequences to discard from these videos.

As a result, the victim sequences had to be obtained from the most popular videos where the concentration of clients was increased due to the increase caused on ZC. Due to this higher concentration, the clients got closer to each other. Consequently, it was necessary to utilize small TW values to make a distinction between the densities of clients observed within each time window.

Conversely, as ZC is reduced, part of the clients no longer request access to the videos of high popularity and instead begin to request the less popular videos. So, the concentration of clients on the less popular videos becomes small, but large enough to allow the formation of sequences that will feed the replacement process. In this context, since the clients stay separated by a larger number of blocks, the size of the time window needs to be expanded to accommodate a greater quantity of clients per window, which causes an imbalance on the number of clients present in each window.

Another way to modify the concentration of clients is shown in Fig. 10c, which presents the correlation between the size of the time window and the number of videos offered for access. Considering that IA and ZC were kept constant for these simulations (since, as demonstrated, these parameter can directly affect the concentration of clients), the increase of the NV caused the clients to be scattered among the available videos in such a way that the number of clients present in each video became smaller. Thus, as NV is increased, it is necessary to enlarge the value for TW in order to create a distinction regarding the number of clients present in the windows that precede each sequence stored in the memory.

Aside from the need to adjust the size of the temporal window to follow the change in clients density in each video (as occurs when the parameters IA, ZC, or NV suffer a variation), Fig. 10d shows that the adjustment of the TW parameter can also be influenced by the bandwidth associated with the link that connects the proxy to the main server. As the link becomes narrower, the algorithm needs to work with a wider time window in order to identify sequences that once cached in the proxy will enable the service to be provided to a larger number of clients. This occurs because, in scenarios with low NB, the CARTE must choose the sequences with better cost-benefit in the longer term, since there are not enough resources on the link to meet all different levels (amplitudes) of demand existing in the shorter term.

On the other hand, as the proxy-server link becomes wider, our algorithm works more efficiently with smaller window sizes since, under these conditions, it is possible to apply additional bandwidth to prioritize simultaneously a larger amount of demands that are high in the short term, but not necessarily in the long term. Thus, the algorithm achieves a high scalability by maximizing not only the use of network but also the use of processing and storage resources to increase the system productivity.

Similarly, Fig. 10e shows that due to the increase in memory size, TW needs to expand to provide a better performance to the proxy. This happens because with the use of a larger memory, the CARTE may invest in sequences that, even not having a higher demand in the short term, produce greater efficiency in the longer term. Therefore, the larger memory not only permits the allocation, but also the preservation of these sequences for a long enough time to extract the benefits resulting from the caching of this content.

Finally, Fig. 10f shows that the TW was little influenced by most video lengths used in our simulations. The most significant variation on TW occurred when VL was increased from 90 to 120 min. In this case, it was necessary to increase TW to better identify the sequences that produce greater efficiency in the longer term, since, due to the increase caused on VL and the preservation of the continuous inflow of clients, a larger quantity of sequences formed in each video began to dispute the spaces available in memory.

However, the subsequent increases in VL did not produce the same result in TW. This occurred because although it can be expected that subsequent additions on VL could generate a proportional increase on TW, the capacity of the memory available in the proxy would not be large enough to store these sequences for such a long time interval. Thus, the best configuration for TW remained roughly stable along the subsequent changes on VL since this configuration maintained a better proportion with the size of memory used in these experiments.

Procedures to configure CARTE

The strategy currently in use to adjust the TW parameter is based on the use of an explorer algorithm, called TW Space Scanner (TSS), which executes a small number of simulations of CARTE for each one in which TW is varied until the better configuration for a target scenario can be found.

TSS works in three steps. In the first step, it carries out a coarse-grained scanning on a set of samples of TW obtained from an initial interval (for example, from 1 to 300). In this scanning, each simulation uses a different sample which is selected through the sum of the increment step (*coarse_step*) with the previous sample value, such that the first sample corresponds to the first value of TW present in the initial interval.

Based on our experiments, we have identified that when *coarse_step* = 20 TSS can detect false maximum points on the curve of efficiency (resulting from the hit ratios produced by the values of TW existing in the initial interval). So, when this configuration is used, the top of the curve of efficiency is clearly highlighted in relation to the other points that belong to the initial interval.

After the coarse scanning, the sample of TW which results in the greatest hit ratio is used as an initial reference to execute the second step that consists of a fine-grained scanning carried out as Fig. 11 illustrates.

Figure 11a shows that TSS uses a fine variation step (*fine_step* = 5) to calculate, from the reference point produced by course scan, the values of L and R that limit by

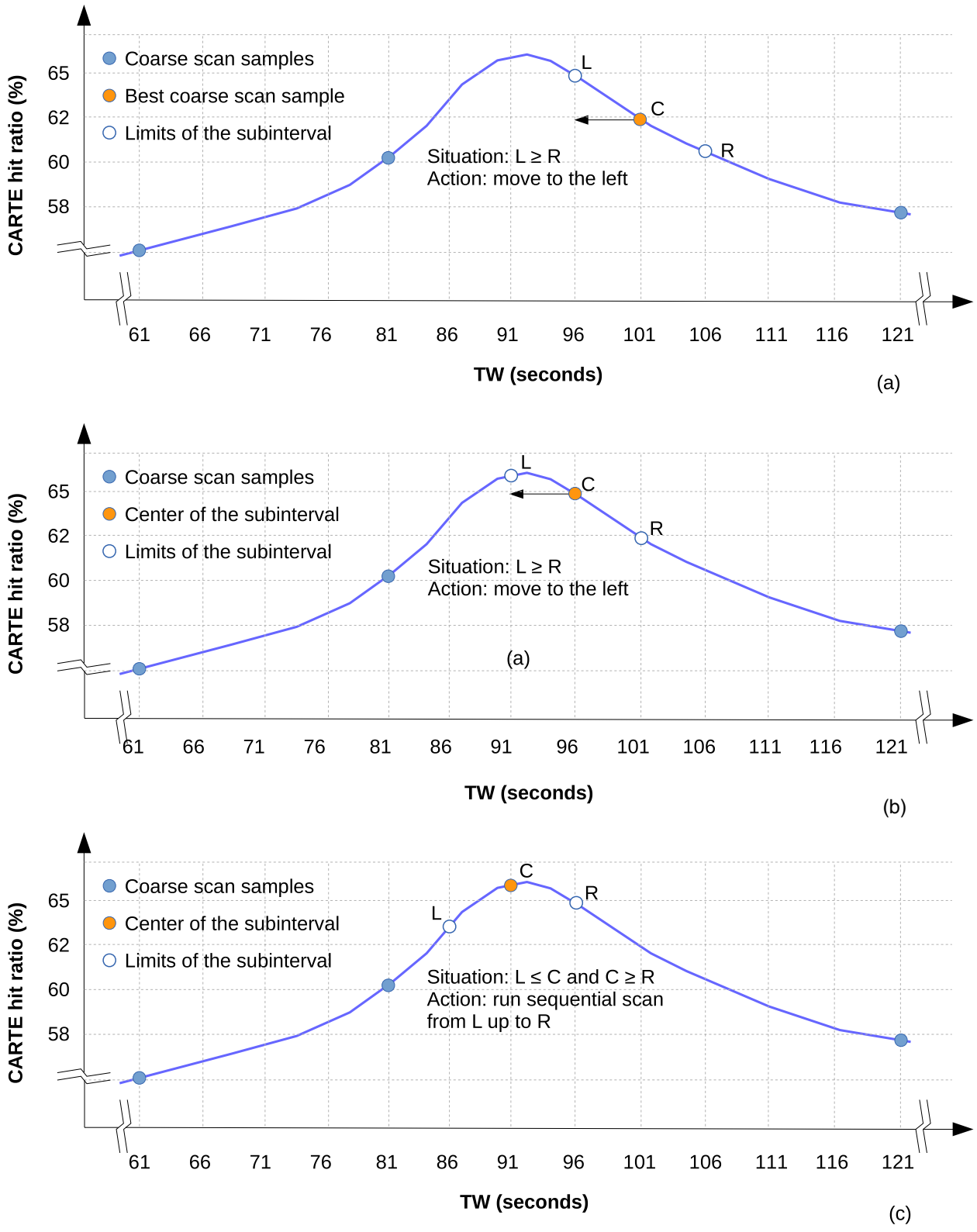

Fig. 11 Functioning of fine scanning. **a** Initial state with the definition of the first values to L, C, and R. **b**, **c** Iterations of the algorithm with displacement of L, C, and R. **c** Final state with L ≤ C and C ≥ R

the left and right, respectively, a sub-interval that has C as the center. After that, as shown in Fig. 11b, c, the algorithm iterates, displacing L, C, and R to one of the sides using the fine variation step until $L \leq C$ and $C \geq R$. The displacement happens in the direction to the lateral limit that presents a greater hit ratio in comparison with the center of the interval.

When the stop condition is achieved (Fig. 11c), the algorithm begins its last step by carrying out a sequential scanning (using an increment step equal to 1) in the sub-interval limited by L and R. As a result of the sequential scanning, TSS identifies the most efficient TW value for CARTE in the target scenario.

Equations 5, 6, and 7 describe how the calculation of the maximum number of simulations is done respectively to the first, second, and third steps of TSS algorithm. The variable *max_TW*, used in Eq. 5, informs the maximum length of TW to the initial interval that goes from 1 until *max_TW*. For the existing scenarios in the functioning range specified in Table 2, *max_TW* tends to not assume values higher than 300. Under these conditions, the execution of TSS produces 30 simulations until it finds the ideal value of TW for the target scenario.

$$\text{coarse\_runs} = (\text{max\_TW} / \text{coarse\_step}) + 1 \quad (5)$$

$$\text{fine\_runs} = \text{coarse\_step} / \text{fine\_step} \quad (6)$$

$$\text{sequential\_runs} = (\text{fine\_step} * 2) - 2 \quad (7)$$

We have been carrying out some more profound researches on the impacts produced on TW by the variation of load parameters. From this, we hope to create a model that allows estimating the best approximate value to TW to a scenario of interest. Doing this, we planned to measure the benefits provided by the substitution of coarse scanning, currently executed by TSS, by a description in software of this model. Finally, we intend to derive from this modified version of TSS a solution to dynamically adjust TW, thus enabling CARTE to reconfigure itself in the face of the possible fluctuations in the workload.

Conclusions

This paper presents the video caching algorithm named Current demAnd Rather Than futurE (CARTE), intended to be used in proxies for video on demand. This algorithm was designed in the scope of a new paradigm in which the caching choices are based exclusively on the observation of the current positioning of the active clients of the system in order to calculate the specific demand that these clients will create for each video chunk in the future. Contrasting our strategy to other similar approaches, the decisions made by our algorithm takes into account only the number of clients positioned inside a time window located in front of each video chunk, thus preventing those clients that are distant of a given chunk to interfere on the calculation of the caching priority for that chunk. Additionally, the size of this time window is the key parameter for configuring the CARTE to achieve its maximum performance in each scenario.

Since the number of active clients within each time window may vary during the operation of the proxy, due to the constant inflow and outflow of clients, our caching algorithm must continually (in the same I/O frequency of the clients) assess which video chunks have higher demand in order to define which portions of the collection must remain stored in the proxy memory to maximize the system productivity.

Consequently, unlike the previous approaches which tend to under-utilize the bandwidth supported by the memory and the processor of the proxy in order to increase the efficiency in the longer term, the CARTE algorithm utilizes these resources more actively in order to provide scalability to support the high demand which usually falls on the most popular videos of a collection. Thus, it provides greater efficiency to the proxy, creating subsequent positive impacts on the implementation costs of VoD system.

To obtain the knowledge of resource consumption practiced by our algorithm and evaluate its performance in terms of the most common metrics, we created a new simulator named PROxy SIMulator (SIMPRO). Our simulator enables a more complete performance analysis with respect to the execution of a video caching algorithm on a particular target architecture, therefore allowing to identify the main physical bottlenecks created by the increase in demand. This ability makes the SIMPRO the first simulator dedicated to the analysis of the VoD proxy performance from the architectural point of view.

Comparative results obtained through this simulation environment show that our new video caching algorithm is capable of achieving a significantly higher hit ratio (17.3–20.7 % of difference at the peak) consuming a moderate quantity of additional resources (10.1–12.6 %), in comparison to other existing algorithms also analyzed.

Complementarily, the analysis of the resources consumed by this algorithm (as well as by the other algorithms also analyzed) revealed that the processor tends to be the major bottleneck of the application when demand increases, consuming in average about two to three times the time spent by the memory to perform the necessary tasks. This allows us to conclude that a new hardware and software partitioning specifically dedicated to our system tends to significantly improve its efficiency.

In future research, we intend to conduct a review of the VoD proxy architecture, as a way to plan efficient

hardware/software mappings from the cost-benefit point of view. Thus, we expect to increase the scalability of the infrastructure available for VoD today and, at the same time, reduce investments for the deployment of this service.

Additionally, since a VoD proxy is frequently designed to operate on continuous mode and the loading conditions can vary during the service provision, we plan to implement the functional requirements to provide our algorithm the ability to perform incrementally the dynamic adjustment of the size of its time window. In this way, the CARTE will be able to adapt itself to every circumstance, thus preserving its maximum efficiency even in the face of the fluctuations on the workload.

To add this new ability on the algorithm, we will use the guidelines presented in this paper, most of which are based primarily on the observation of the density of clients (number of clients per time interval) caused by the proxy workload conditions. Once this density has direct impacts on choosing the best size for the time window, the key aspect for the dynamic adjustment of the algorithm is to monitor the workload to incrementally produce changes on the size of the time window, in order to find the optimal settings for the system.

Endnotes

[1]As opposed to distributed approach also commonly found in the literature [49].

[2]Clients connected to the proxy for receiving a video stream.

[3]Not more than 300 seconds of video in our standard simulation scenarios.

[4]One second of video per client in our experiments.

[5]Corresponding to the availability, in the memory of the proxy, of the requested video blocks.

[6]Thirty simulations were executed obtaining a confidence of at least 95 % to a confidence interval less than 1 % of the measurement value.

[7]Equivalent to 3,480 rounds of service executed or 3,480 Mbits of video transmitted at 1Mbps (considering the transmission to the first client to become active in the system).

[8]As a partial criterion, along with the size of sequences.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
BSN and AAS (BSN's advisor) conceived the objectives and methodology to conduct the research which is being developed in the context of BSN's doctoral in the microelectronics program at the Federal University of Rio Grande do Sul (UFRGS) in the south of Brazil. BSN is mainly responsible for the idealization and design of the contributions presented in the paper, also working on the implementation of the algorithms and the simulator, as well as on the collection and interpretation of the data and in the paper writing. ASV worked closely with BSN coding the software and providing technical support for the interpretation of results obtained from the simulations. AAS contributed with important observations to technically improve the implementations and the presentation of the work. All authors reviewed the final version of paper and agreed with its submission to the Journal of the Brazilian Computer Society.

Acknowledgements
This research is financially supported by the National Council for Scientific and Technological Development - CNPq (Process number: 554064/2010-3) and the Federal University of Pampa - UNIPAMPA (Project number: 02.070.10).

Received: 9 June 2014 Accepted: 21 June 2015
Published online: 03 August 2015

References

1. Cisco (2011) Visual networking index: forecast and methodology, 2011-2016. Cisco, Jose, California, United States of America. http://www.itu.int/md/dologin_md.asp?id=S12-WTPF13IEG2-INF-0002!!PDF-E. Accessed 19 Jun 2012
2. Ma KJ, Bartoš R, Bhatia S (2011) Review: a survey of schemes for internet-based video delivery. J Netw Comput Appl 34(5):1572–1586
3. Passarella A (2012) Review: a survey on content-centric technologies for the current internet: Cdn and p2p solutions. Comput Commun 35(1):1–32
4. Pathan M, Buyya R, Vakali A (2008) Content delivery networks: state of the art, insights, and imperatives. In: Lecture Notes Electrical Engineering. Springer, Berlin. pp 3–32
5. Vinay A, Prakash A, Kumar DSK, Nagabhushan K, Anitha TN (2011) A multithreaded based load balancing framework for video-on-demand systems. In: Proceedings of the International Conference & Workshop on Emerging Trends in Technology. ACM, New York. pp 363–369
6. Li J, Chen Z (2009) Sliding-window caching algorithm for streaming media server. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human. ACM, New York. pp 1152–1159
7. Chiu H, Chang CH, Tseng CW, Liu CS (2011) Window-based popularity caching for IPTV on-demand services. CN 2011:32–323232
8. Li J, Li F, Jiang X (2011) Flexible-segmentation-jumping strategy to reduce user-perceived latency for video on demand. Appl Comp Intell Soft Comput 2011:1–118
9. Famaey J, Iterbeke F, Wauters T, De Turck F (2013) Towards a predictive cache replacement strategy for multimedia content. J Netw Comput Appl 36(1):219–227
10. Yu J, Chou C, Yang Z, Du X, Wang T (2006) A dynamic caching algorithm based on internal popularity distribution of streaming media. Multimedia Syst 12(2):135–149
11. Yu J, Du X, Wang T, Tung Chou C (2006) Internal popularity of streaming video and its implication on caching. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 01. IEEE, Los Alamitos. pp 35–40
12. Li P, Zheng W, Zhang K (2011) The design of streaming media proxy server based on patching first algorithm. In: Strategic Technology (IFOST), 2011 6th International Forum On. IEEE, Los Alamitos Vol. 2. pp 643–647
13. Chen H, Jin H, Sun J, Liao X, Deng D (2003) A new proxy caching scheme for parallel video servers. In: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing. ICCNMC '03. IEEE Computer Society, Washington, DC, USA. p 438
14. Ishikawa E, Amorim CL (2009) Collapsed Distributed Cooperative Memory for Interactive and Scalable Media-on-demand Systems. U.S. Patent 7,596,664, to COOPE/UFRJ, 29 Sept 2009
15. Hong D, De Vleeschauwer D, Baccelli F (2010) A chunk-based caching algorithm for streaming video. In: NET-COOP 2010 - 4th Workshop on Network Control and Optimization. Gent, Belgique, INRIA, Grenoble. pp 33–34
16. Wu T, De Schepper K, Van Leekwijck W, De Vleeschauwer D (2012) Reuse time based caching policy for video streaming. In: Consumer Communications and Networking Conference (CCNC), 2012. IEEE, Los Alamitos. pp 89–93
17. Tu W, Steinbach E, Muhammad M, Li X (2009) Proxy caching for video-on-demand using flexible starting point selection. Multimedia IEEE Trans 11(4):716–29

18. Kai-Chun L, Yu HF (2012) Adjustable two-tier cache for IPTV based on segmented streaming. Int J Digital Multimedia Broadcast 2012(1). http://dx.doi.org/10.1155/2012/192314
19. Carbunar B, Pearce M, Vasudevan V, Needham M (2011) Predictive caching for video on demand cdns. In: GLOBECOM'11. IEEE, Los Alamitos. pp 1–5
20. Almeida JM, Krueger J, Eager DL, Vernon MK (2001) Analysis of educational media server workloads. In: Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM, New York. pp 21–30
21. Dan A, Sitaram D, Shahabuddin P (1996) Dynamic batching policies for an on-demand video server. Multimedia Syst 4(3):112–121
22. Dhage SN, Meshram BB (2013) Design and implementation of video servers for VOD system. Int J Cloud Comp 2(1):61–88
23. Summers J, Brecht T, Eager D, Wong B (2012) Methodologies for generating http streaming video workloads to evaluate web server performance. In: Proceedings of the 5th Annual International Systems and Storage Conference. ACM, New York. pp 2–1212. http://doi.acm.org/10.1145/2367589.2367602
24. Intel (2005) PCI Express Ethernet Networking. Intel, 2005, Santa Clara. http://www.intel.com/content/www/us/en/pci-express/pci-express-ethernet-networking-paper.html. Accessed 23 Aug 2011
25. Willhalm T (2012) Intel Performance Counter Monitor - A Better Way to Measure CPU Utilization. Intel, Santa Clara. https://software.intel.com/en-us/articles/intel-performance-counter-monitor. Accessed 5 Jan 2012
26. Austin T, Larson E, Ernst D (2002) Simplescalar: an infrastructure for computer system modeling. Computer 35(2):59–67
27. Granado AC Experimental Evaluation of the Collapsed Cooperative Video Cache for Video on Demand Systems. Master's thesis, Federal University of Rio de Janeiro, COPPE
28. Decker C, Riedel T, Beigl M, Krohn A (2006) A file system for system programming in ubiquitous computing. Personal Ubiquitous Comput. 11(1):21–31
29. Jiang C, Yu Z, Jin H, Xu C, Eeckhout L, Heirman W, Carlson TE, Liao X (2013) Pcantorsim: accelerating parallel architecture simulation through fractal-based sampling. ACM Trans Archit Code Optim 10(4):49–14924
30. Smit M, Stroulia E (2013) Simulating service-oriented systems: a survey and the services-aware simulation framework. Serv Comput IEEE Trans 6(4):443–456
31. Abad P, Prieto P, Menezo LG, Colaso A, Puente V, Gregorio JA (2012) Topaz: an open-source interconnection network simulator for chip multiprocessors and supercomputers. In: Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium On. IEEE, Los Alamitos. pp 99–106
32. Broadcom (2009) 1-Gigabit TCP Offload Engine. Broadcom, Irvine. https://www.broadcom.com/collateral/wp/5709-WP101.pdf. Accessed 5 Jan 2012
33. Netflix Netflix, Los Gatos, California. https://help.netflix.com/en/node/306. Accessed 08 Mar 2014
34. Yu H, Zheng D, Zhao BY, Zheng W (2006) Understanding user behavior in large-scale video-on-demand systems. SIGOPS Oper Syst Rev 40(4):333–344
35. Adhikari VK, Guo Y, Hao F, Varvello M, Hilt V, Steiner M, Zhang ZL (2012) Unreeling netflix: Understanding and improving multi-CDN movie delivery. In: INFOCOM, 2012. IEEE, Los Alamitos. pp 1620–1628
36. Chan S-HG, Xu Z, Liu N (2013) Optimizing video-on-demand with source coding. In: Multimedia and Expo (ICME), 2013 IEEE International Conference On. IEEE, Los Alamitos. pp 1–6
37. Ji W (2013) The design of passive optical networking+ethernet over coaxial cable access networking and video-on-demand services carrying. Fiber Integrated Optics 32(4):268–279
38. Campanotti B, Hurt A (2013) Building real world media in the cloud. Motion Imaging Journal 10:1–7
39. Dewangan A, Jalihal D (2013) Statistics based energy efficient caching decisions for IPTV services. In: Communications (NCC), 2013 National Conference On. pp 1–5
40. Netflix All DVDs Releasing This Week. Netflix, Los Gatos, California. http://dvd.netflix.com/AllNewReleases?lnkctr=NavAllNewReleases. Accessed 08 Mar 2014
41. Avramova Z, Wittevrongel S, Bruneel H, De Vleeschauwer D (2009) Analysis and modeling of video popularity evolution in various online video content systems: power-law versus exponential decay. In: Evolving Internet, 2009. INTERNET '09. First International Conference On. IEEE, Los Alamitos. pp 95–100
42. Dan A, Sitaram D, Shahabuddin P (1994) Scheduling policies for an on-demand video server with batching. In: Proceedings of the Second ACM International Conference on Multimedia. ACM, New York. pp 15–23
43. Qudah B, Sarhan NJ (2010) Efficient delivery of on-demand video streams to heterogeneous receivers. ACM Trans Multimedia Comput Commun Appl 6(3):20–12025
44. Ryu M, Kim H, Ramachandran U (2011) Impact of flash memory on video-on-demand storage: Analysis of tradeoffs. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems. ACM, New York. pp 175–186. http://doi.acm.org/10.1145/1943552.1943577
45. Bataa O, Lamjav E, Batsuuri S, Purevdorj U, Naimannaran C, Kim YI, Gonchigsumlaa K (2012) Service control algorithm of providing efficient video-on-demand service using hybrid mechanism. In: Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference On. IEEE, New York. pp 3507–3512
46. Li J, Yang J, Xi H (2009) A scalable and cooperative caching scheme in a distributed VOD system. In: Communication Software and Networks, 2009. ICCSN '09. International Conference On. IEEE, Los Alamitos. pp 247–250
47. Ishikawa E, Amorim CL (2003) Collapsed cooperative video cache for content distribution networks. In: In Proceedings of the Brazilian Simposium on Computer Networks (SBRC). SBC, Porto Alegre, Brazil. pp 249–264
48. Jung J, Krishnamurthy B, Rabinovich M (2002) Flash crowds and denial of service attacks: characterization and implications for CDNS and web sites. In: Proceedings of the 11th International Conference on World Wide Web. ACM, New York. pp 293–304
49. Zeng Z, Veeravalli B, Li K (2011) A novel server-side proxy caching strategy for large-scale multimedia applications. J Parallel Distrib Comput 71(4):525–536