Workload Characterization for News-on-Demand Streaming Services

Frank T. Johnsen and Trude Hafsøe
Department of Informatics
University of Oslo, Norway
{frankjo, truhafso}@ifi.uio.no

Carsten Griwodz and Pål Halvorsen
Networks and Distributed Systems
Simula Research Laboratory, Norway
{griff, paalh}@simula.no

Abstract

This paper focuses on design issues for multimedia distribution architectures and the impact workload characteristics have on architecture design. Our contribution is an analysis of server load and user behavior in a News-on-Demand environment, with focus on access patterns, popularity modeling, and the formation of traffic peaks. Finally, we evaluate an existing synthetic workload generator, MediSyn, and suggest some enhancements which will improve its suitability for News-on-Demand workload modeling.

1. Introduction

Designing an architecture for the dissemination of large amounts of multimedia data requires knowledge of the content itself and the way users interact with the content. Such knowledge enables the system architect to consider media and use, and thus employ the proper techniques when designing the system. This encompasses choices of hardware and provisioning resources such as storage, network, file system, page caching algorithm and processing power. Further enhancements such as the use of caching proxies can be employed to reduce the load on the origin server and also limit bandwidth consumption. An optimized architecture can provide prompt responses to a large number of users while ensuring the cost-effectiveness of the system. Overprovisioning resources will ensure good service to users, but initial deployment costs are high.

Workload characterization is an important tool in designing architectures for large scale multimedia distribution, such as content distribution networks (CDNs). Statistical knowledge of user access patterns enables the architect to dimension the system accordingly. This is important for a single server system, but even more so in hosting centers where several independent services share the resources. In shared systems, knowledge of the diurnal access patterns of the hosted services can be used to ensure efficient resource usage, while simultaneously avoiding system overload due to aggregated traffic.

News-on-Demand (NoD) is one service that relies on such distribution systems and that is becoming increasingly popular. In Norway today, we see that online newspapers are gaining more and more users and that paper editions are losing ground [1].

We have obtained two years worth of logs of streaming media accesses from *Verdens Gang* (VG), Norway's largest online newspaper [1]. We performed an initial analysis of these logs in [2], and have now performed additional investigations of access patterns, stream interactions, lifetime and popularity of streams and traffic peak formation. This paper focuses on the aspects of NoD workload characteristics which may affect CDN architecture design:

- **Server and network resource usage**
  To properly dimention the system, one needs to estimate the number of concurrent users that are to be expected, and their interaction with the content. We present our findings relating to bandwidth, server memory working set, and more; these aspects are affected by access patterns, popularity and interactivity.

- **Diurnal access patterns**
  The diurnal access pattern defines the variation in access intensity for a specific period of time. This pattern is important for modeling the variations in

resource usage within a given time period, for example a day. In a single server system, one has to provision for the highest expected load, while shared systems can use this knowledge to perform statistical multiplexing of the resources. The diurnal access pattern varies from culture to culture as a result of social differences. We compare our findings to a similar study from Spain.

- **Popularity**
  Knowledge of popularity is important for modeling server and network load. Traditionally, a Zipf distribution has been used to model file popularity for Video-on-Demand (VoD) workloads, but we show that it does not accurately model a news service with an open-ended archive and note some flaws in its use. We discuss the method used for popularity generation in the synthetic workload generator MediSyn [3] and relate it to our data set.

- **Interactivity**
  While file popularity concerns the total number of accesses to each file, the interactivity relates to the aggregated client interactions within a file. Thus, an analysis of client interactivity can give us an overview of how the users use the material and which parts of the files are most popular. Such knowledge is particularly useful when considering whether to include caching solutions as part of the system architecture. For example, if a high percentage of incomplete accesses are concentrated on the beginning of most files, then employing prefix caching [4] may be a good idea.

- **Traffic peaks**
  Some days have an access intensity far surpassing the average. These peaks put high loads on the server and demand a lot of bandwidth from the network. We have investigated how such peaks form and show that a *hyperbolic tangent* function can be used to describe them. If a system can detect the formation of a peak it may have time to react and allocate the resources needed to accomodate it. Our investigations show that peaks form over a number of minutes, thus leaving the system with ample time to react.

The remainder of the paper is organized as follows: A brief overview of related work is given in Section 2. Next we present the workload characteristics that have been derived from the logs in Section 3. In Section 4 we discuss an existing synthetic workload generator in the context of our results. Finally, Section 5 concludes the paper.

2. Related work

In a previous study [2], we performed an initial analysis of some aspects of NoD workloads. That poster paper discusses access intensity, popularity modeling using the Zipf distribution, user interaction with streams and the existence of short term effects. In this paper, we present further findings from our continued studies of the same log material.

Kim et al. [5] analyzed logs from an online newspaper in Korea. They argue that NoD is different from VoD in terms of media type and size, life cycle of articles and frequency of users' interaction. They develop a popularity model called the Multi-Selection Zipf distribution and show how one can perform prefetching and placement between disk and cache based on this model.

Pañeda et al. [6] have investigated user behavior of a VoD service with a wide variety of subjects and lengths, followed by a study of server workload in a large online newspaper [7]. Their server workload analysis was performed using Spanish logs, and we compare our findings from the Norwegian newspaper with their results.

MediSyn [3] is a generator for synthetic multimedia workloads developed by Hewlett-Packard (HP). It provides a valuable tool for evaluating systems based on its synthetic traces. The generator is developed based on a workload study performed on logs of company internal traffic from HP's enterprise servers.

Yu et al. [8] present a measurement study of a large VoD system in China. They found that changes in video popularity is strongly influenced by the introduction of new content as well as external factors. They point out that the study which forms the basis for MediSyn may have limited applicability due to the limited choice of topics in the system, but do not attempt to prove or disprove this in any way. In our study, we compare our results with the modeling techniques in MediSyn, evaluating its suitablity for model-

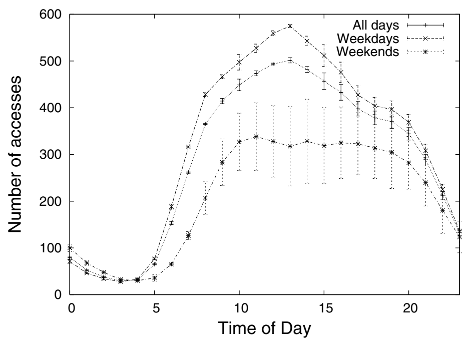


Figure 2. Hour-to-hour access intensity

ing NoD workloads.

Rocha et al. [9] investigate alternative mechanisms for scalable streaming to interactive users. They identify a set of workload aspects that impact the scalability of classes of streaming protocols. They propose and evaluate several optimizations to streaming protocols and show that it is possible to reduce the average server bandwidth required for interactive workloads. One significant contribution of that study is their classification of interactivity into three categories; low, medium and high interactivity based on client interaction patterns. In our study, we compare our findings with their criteria.

3. Workload characteristics

The logs contain 4.6 million client accesses to about 3,500 different files, of which just under 1,000 are audio files. When we use the term access in our analysis discussions, *one access corresponds to one successful request line in the log*. We have removed all failed requests from the logs.

The files in the data set vary in size from a few KB and up to several MB. Video files make up the majority of the total size of the files, with roughly 17 GB compared to the 284 MB of audio. Audio files are more uniform in size than video, most likely because the majority of audio files are hitlist music files, which tend to be more or less the same length.

3.1. Access Intensity

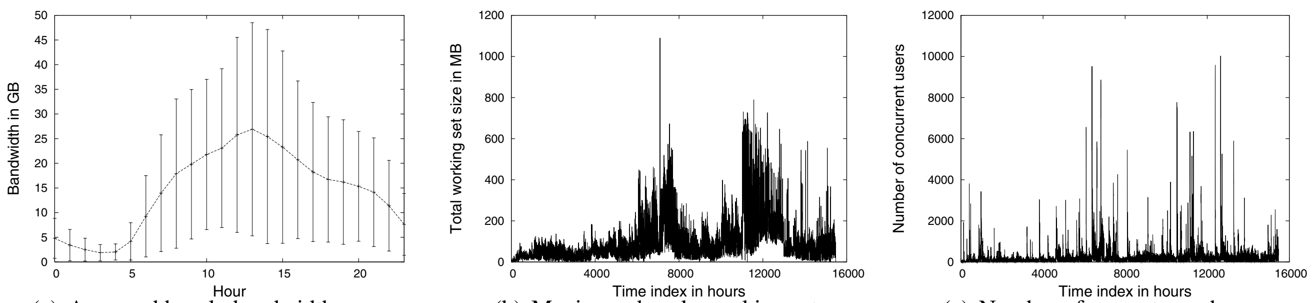
Load can be measured by investigating bandwidth usage, memory usage and/or the number of accesses. We analyze all these three aspects.

The streaming media, being both audio and video files, have quite different ratios when it comes to hits and bandwidth. Even though audio has quite a few hits, video streaming constitutes the bulk of the bandwidth consumption. Our analysis shows an average streaming bandwidth of 14.43 GB/hour or 4.1 MB/s. During the busiest time of day the bandwidth consumed by streaming traffic often exceeds 25 GB/hour. Figure 1(a) shows the average hourly bandwidth usage with standard deviation of the VG streaming server for the entire two year period. Monthly averages show that the highest hourly bandwidth usage recorded there is 94.92 GB/hour.

Another way of defining server load is by investigating memory usage. The amount of memory used on the server is determined by the set of files currently being accessed by clients. For a streaming server the memory required to hold this set of files can be called the *working set* [7]. The maximum working set for the duration of the logs was 1,090 MB. Using an interval of one hour, the interval average working set was 44 MB. This indicates that it may be beneficial to utilize a memory caching technique such as generalized interval caching [10]. By using such a technique to optimize memory usage the number of hard drive accesses will be kept at a minimum. The maximum hourly working sets for the entire log span is shown in Figure 1(b).

Variations in the number of concurrent users will affect the load on the server. The working set depends on the number of unique files being accessed. The access count, on the other hand, indicates the number of concurrent streams from the server. Figure 1(c) shows the variations in accesses from hour to hour. A large number of users accessing the same file will result in a small working set, but a high access count and a correspondingly high bandwidth consumption. Thus, the working set and the access count are both important and orthogonal aspects of system load. Respectively, they determine the amount of memory needed and the total bandwidth required.

The access count, like the bandwidth usage, follows a regular daily pattern. News requires interactivity from the users. This means that diurnal access patterns, and thereby server load, depend largely on the daily habits of users, and will vary from one culture to another. [6] reports that the user access pattern



(a) Averaged hourly bandwidth usage (b) Maximum hourly working set (c) Number of requests per hour

Figure 1. Server load over time

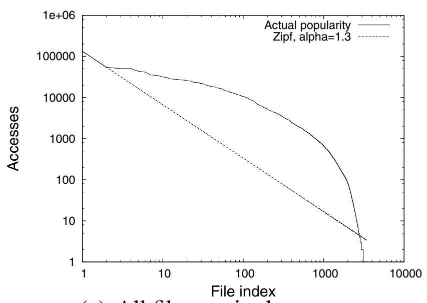
experienced by a Spanish news service clearly shows how users have a long midday lunch break, and that users are most active during the evening. Norwegian social habits are different from those experienced in Spain; most Norwegians work from 8 am to 4 pm, and these working hours are strictly adhered to by the majority of employees. This pattern is clearly visible in Figure 2, which shows the average number and standard deviation of accesses for all days, weekdays and weekends, respectively. During workdays the main bulk of accesses occurs during working hours, with the highest point being reached around noon. During weekends the total number of accesses is significantly lower, and the standard deviation is higher. In addition, users start using the news service later in the day during weekends. For a detailed discussion of the reasons for variations during the weekend, see our previous study in [2].

3.2. Modeling popularity with the Zipf distribution

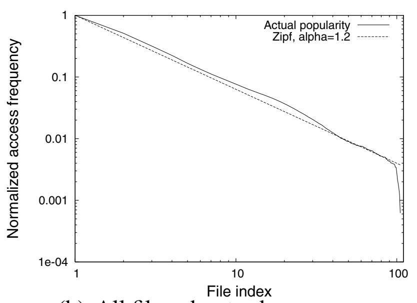
The Zipf distribution is used to describe the popularity distribution of multimedia contents in various applications, among others VoD scenarios. It is not clear whether this assumption of Zipf-distributed popularities holds for news items in a NoD scenario. Cherkasova and Gupta [11] note for an enterprise media server workload that workloads do not necessarily follow the Zipf distribution on a longer time scale, while they do on short time scales. Like the authors of many other papers, they observe a common problem with the Zipf distribution. It is a hyperbolic function and not additive. When you cut off part of a Zipf function, for example by handling highly popular items through caching and observe only the remaining tail, that tail can not follow a Zipf distribution. In fact, if the original data followed a Zipf distribution, then the tail can not be fitted correctly to a Zipf distribution for another skew factor.

Furthermore, if a popularity distribution can be matched by a Zipf distribution for one skew factor, then this approximation may hold for as long as the popularity of all observed items remains unchanged. If the popularity of items develops over time, then the relative popularity of an item differs between its short term relative popularity and its long term relative popularity; if a day's top news item is replaced by another one every day, then the long term relative popularity of these news items can not be Zipf-distributed. This is not considered in [6], where the authors rather try to find a limit value for the skew factor when the time (and the number of archived news items) goes toward infinity. Kim et al. [5] propose a "new popularity model called the Multi-Selection Zipf distribution". The semantic problem of identifying model and distribution nonwithstanding, they use the one-month average popularity of news and try to match this with a Zipf distribution although the relative popularity of news items is known to change rapidly. Predictably, the outcome is that popularity differences at the top are less pronounced than a Zipf distribution would express. The authors observe correctly that popularity spreads over several articles (in the course of a month), but the conclusion drawn is not that the granularity of the observation is too coarse, but that articles should be grouped together. Critically, this was done without considering the temporal correlation of those articles.

Such problems can be avoided by considering only phases that are so short that the popularities are stable, as in [12]. To exploit the data set from a dura-

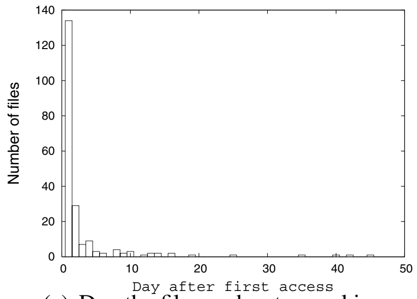


(a) All files; entire log span

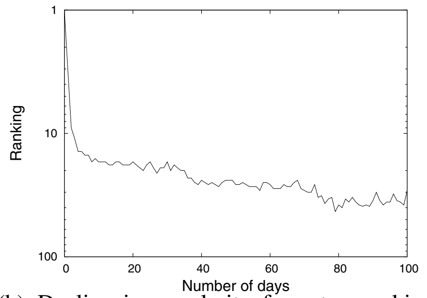


(b) All files; day-to-day average

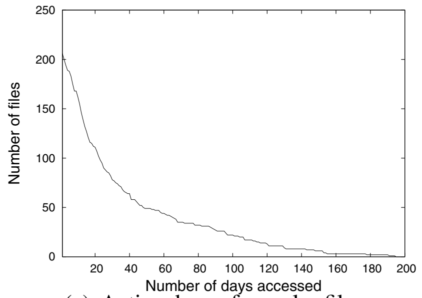
Figure 3. Popularity distribution using Zipf



(a) Day the file reaches top ranking



(b) Decline in popularity from top ranking over time



(c) Active days of popular files

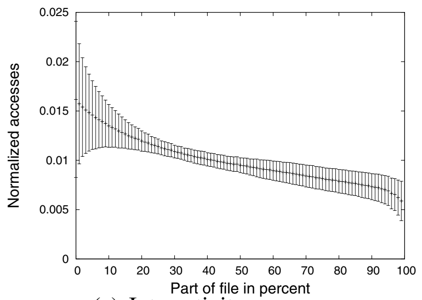
Figure 4. Long-term popularity development

tion spanning several stable phases, the items for each phase can be sorted by popularity and the values averaged for every index among all of the stable phases. If every single stable phase can be approximated by a Zipf distribution, then it is also possible to approximate the averaged short term period to this distribution function.

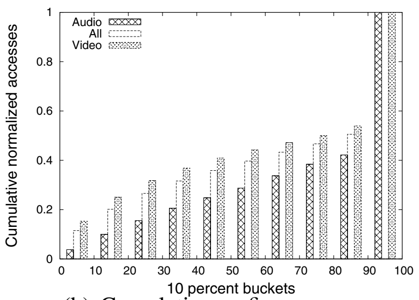
However, Almeida et al. [12] do not observe a Zipf distribution in spite of considering only stable phases. They observe a different popularity distribution that they describe as a concatenation of two Zipf-like distributions. Obviously, the popularity distribution of their content is extremely heavy-tailed. The data presented by Yu et al. [8] shows the same properties, and so does our data: The entire log period as a phase is shown in Figure 3(a). The picture is totally different when we consider the duration of the stable phase. Figure 3(b) shows the results for an average where the phase is a day. The Zipf distribution, here plotted with a skew factor of 1.2, does not quite fit; there is a heavy tail, but it describes the top movies quite well. Yu et al. attribute the heavy tail to old videos that remain in the VoD system that they investigated. Zipf describes also our news popularity very well for nearly all content in the system. Our investigation also shows how important it is to know the duration of the stable phase of the content of an on-demand system before employing the Zipf distribution to describe its popularity distribution.

3.3. Long-term popularity development

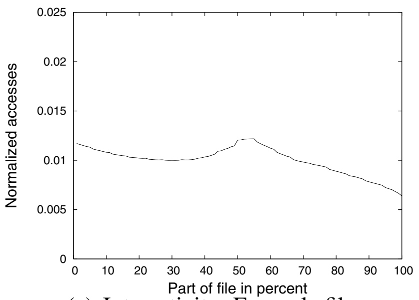
The most popular files make up a large percentage of the load experienced by the streaming server. The manner in which users interact with these files can therefore have a large impact on the service, and a better understanding of how these popular files are used is needed. In the following analysis, popular files are defined as *the collection of files which topped the daily hit statistics at least once*. There are three aspects of popular files that are of particular interest, namely how fast a newly published file achieves the top ranking, how long popular files remain popular and for how long they are accessed at all. Figure 4(a) shows that the majority of popular files reach top ranking on the same day as they are published. This is a consequence of the nature of news, where the breaking news is presented first on news websites and is of high interest to most readers. The few files that reach top ranking late in their lifespan are mostly audio files containing music.



(a) Interactivity averages



(b) Cumulative prefix accesses



(c) Interactivity: Example file

Figure 5. File access patterns

Once a file has reached the highest ranking it is likely to drop in popularity shortly after. This is illustrated by Figure 4(b), which shows the decline in popularity over time for these popular files. Most files drop to below tenth place in just a few days. This indicates that more recent news has captured the attention of readers, and older news is less interesting to view. Figure 4(c) supports this. It shows the number of active days of the most popular files, where an *active day* is a day where the file is accessed at least once. For most files, the set of active days is one continuous period, but some files have a few days when they are not accessed at all. Such days are not considered a part of the files' active days, and are thereby not counted. We see that 50% of the popular files are active 20 days or less, whereas a few files have a relatively long period of active days; the longest being 200 days. Only one file was accessed for that long.

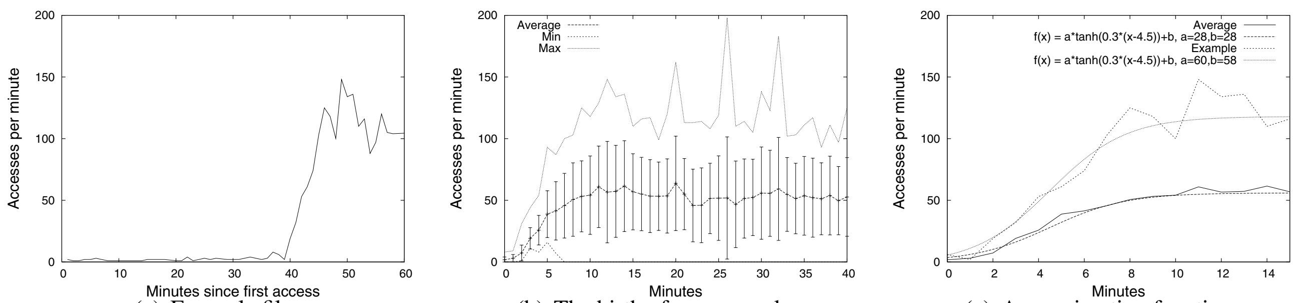
3.4. Stream Interaction

Users of the news service can interact with the news streams using VCR-like commands such as pause, stop, fast forward and rewind. In addition, they can also make jumps into a video stream and then start playback, not at the start of the file, but rather further into the stream. Not all of these interactions are visible to the server, the user might for instance pause playback without this interrupting the stream from the server. In this case, the arriving data will be buffered by the client software. Other interactions require some action to be taken by the server, and this is then visible in the streaming logs. The logs also show the offset at which a user starts playback, and the duration of playback, which, together with the file length, can be used to calculate how large a percentage of the file the user views. All interactions visible to the server can potentially affect the performance of the server and must be taken into account when evaluating server load.

Rocha et al. [9] define three interactivity categories for streaming media; High, Medium and Low interactivity workloads. Their characterization is based on average request duration and average request start position. Our data set has an average request duration of 53% of the total media length, and the majority of accesses are for the start of the file. Audio files have an average request duration of 73%, whereas the video files have an average of 45%. This places NoD media in the low interactivity class.

The vast majority of accesses in the logs, 99.3%, are for playback, and the rest is divided between rewind and fast forward. This means that the total server load is determined by playback, and that VCR-like operations have very little impact on it. A more significant factor is partial accesses, which are accesses where the user views only part of the file. Out of the 4.6 million total accesses over 3 million are partial. Jumping into the file, either at the beginning or some later part of the file, is registered as "playback" in the log, with the file offset indicating the start point. Of the partial accesses, around 700,000 accesses (approximately 15%) have an offset greater than zero, meaning that the user jumped and started playback somewhere else than at the beginning of the file. Just over half of all accesses which start at the beginning of the media are aborted before playing to completion.

Figure 5(a) shows the normalized access distribution averaged for all files, including the standard deviation. Accesses for all files are concentrated at the start of the file, and client interest is uniformly declining throughout the duration of the file. This illustrates



(a) Example file (b) The birth of access peaks (c) Approximation function

Figure 6. Traffic peak formation

the fact that most users will start playback at the beginning of the file, and view it either to completion or until they lose interest and stop. Such an access distribution suggests employing prefix caching [4].

Focusing only on the accesses that start playback at the beginning of the file, we found that for video files 34.8% of the users stop playback during the first third of the stream, while most other users do not stop until somewhere in the last third. Very few users, less than 3%, stop viewing video files in the middle of the file. For audio files this is somewhat skewed, as only 6.2% stop playback in the first two thirds combined, while the rest listen to the file until the last third. For a complete overview of this distribution, where the accesses have been divided into 10% buckets, see Figure 5(b).

So far we have seen that on average NoD exhibits low interactivity, and accesses are weighted towards the beginning of the files. There are, however, exceptions to this when looking at single files. Figure 5(c) illustrates the access distribution of the single most popular file in the system. This file exhibits medium interactivity, and the middle part is actually more popular than the beginning. In this case it is easy to explain this anomaly; the file is a video of a car accident, and the most popular part shows the cars crashing. Users who have seen the file return to watch this part again, sometimes more than once. This indicates that considering employing a segmented caching scheme such as *adaptive and lazy segmentation* [13] may be beneficial in a NoD CDN rather than simpler schemes such as prefix caching [4].

3.5. Traffic peak formation

Some days have an access intensity far surpassing the average. Unexpected increases in load can overload the server and network and leave the streaming service unusable. This effect is difficult to avoid, but if the start of the peak formation can be detected automatically by a server before it becomes overloaded, then the CDN might be able to perform resource reallocation or employ techniques such as multicast for the most popular streams in order to alleviate the problem. Peak formation detection is only possible if the peak forms over time. If the peak forms instantly the system will go from normal load to overloaded without being able to handle the problem.

Figure 6(a) shows a peak formation for the single most popular file. After the low, stable access count for the first 40 minutes the file grew a peak up to above 100 accesses per minute during a timespan of less than 10 minutes. This may be due to a slashdot effect that appears when the story belonging to this clip appears on the web page. We can not verify this since we do not have the corresponding web logs.

We have investigated how the access count of files causing peaks increases over time. In order to detect peak formation, one requires statistical knowledge of peaks. We employed a simple test; summing up the count of accesses to each file from minute to minute, and checking when the slope was greater than 1 for two consecutive minutes. This caught all the peaks, but also gave a lot of false positives. Thus, we need a more complex way to detect peaks based on statistical knowledge of peaks. To ignore the false positives, we limit our investigations to files with over 10,000 accesses on a given day, and we define the start of a peak to be *the first time the number of accesses per minute is greater than 10.* Some files have a low but stable access pattern, so we do not consider files with a maximum number of accesses per minute less than 20 overall. The time from the first access to a file un-

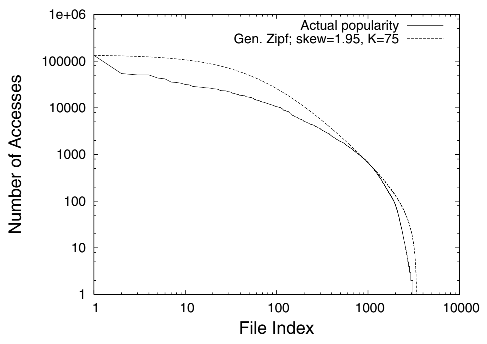


Figure 7. Generalized Zipf-like distribution according to MediSyn step 1

til peak formation begins differs widely between files. From the first access to a file, the shortest time until a peak started forming was 2 minutes, whereas the longest was 31 hours. The average time from the first access to a file till the peak started was 2.6 hours, with a standard deviation of 4.8 hours. We calculated the average accesses per minute during each peak, and also the standard deviation. Figure 6(b) illustrates this, along with the minimum and maximum number of accesses per minute. We see that a peak builds fairly rapidly with a massive increase in accesses during the first 5-10 minutes, after which it remains fairly stable for an extended period of time. See Figure 1(c) for an overview of the access peaks experienced by the server during the time captured by our log material. Having performed this study, we found that we can employ a hyperbolic tangent function to describe the average calculated peak, see Figure 6(c). The figure also shows how one can approximate one specific example peak, namely the peak from Figure 6(a) with the same function but using a different set of parameters. The hyperbolic tangent function can be used to model peak growth, and should perhaps also be considered for use in a peak detection scheme. To reduce peak load on the origin server one could consider using caching proxies or predistributing content to replicated servers in a CDN.

4. Workload generation using MediSyn

Trace driven evaluation is a well known tool for investigating the usability of various content distribution techniques, for example evaluating the performance of caching algorithms. Such evaluations are useful for optimizing an existing service. When building a completely new service, or when expanding an existing service, the traces needed do not exist. In these cases one needs to use a synthetic workload that is representative for the service type and expected network and server load.

MediSyn [3] is the most sophisticated such tool currently available that is known to us. We evaluate the usefulness of MediSyn for NoD workload generation by comparing our log analysis results with the assumptions that MediSyn is based on. It generates accesses in a complex eight-step process:

1. **Popularity**
   The object popularity is generated by a generalized Zipf-like distribution that is capable of capturing the circular curve popularity distribution observed in systems with open-ended archives. We compare the synthetic generalized Zipf-like distribution employed by MediSyn to the VG log material in Figure 7, and see that it fits fairly well.

2. **Duration**
   This step generates the duration of all objects. A random algorithm is used to match duration with frequency. This is based on the observation that there is no correlation between file duration and access frequency, a fact that we tested and found true also for the files in our logs.

3. **Prefix**
   Determining the prefix distribution for each file: The ratio of complete versus incomplete accesses is calculated, followed by the ratio of partial accesses. Medisyn does not model interactivity, and assumes all accesses start at the beginning of the file. In our study, 33% of the accesses are complete and the remaining 67% are partial. However, only 52% of the accesses are partial accesses starting at the beginning of the file.

4. **Bitrate**
   The bitrates are modeled by randomly choosing a typical encoding rate for each file. This is done because the MediSyn study found a very weak correlation between file size and bitrate. This holds true in their scenario because they have material from a plethora of different publishers. The files on the VG servers, on the other hand, show

a strong correlation between file size and bitrate. This is because VG is the main publisher of the content, and thus uses the same editing and publishing techniques for all content.

5. **File arrival**
   This step generates the file introduction day, the number of files introduced per day, the introduction time and the gap to the introduction of successive files. We are unable to verify this point, since our log material does not contain information about file publication time.

6. **Lifespan class**
   A lifespan class is assigned to each file. This class is either the Pareto distributed *news-like* lifespan class or the lognormal distributed *regular* lifespan class. The *lifespan* of files is the total number of days from the first access to the last access, also counting the days in between which have no accesses.

   From the VG logs we found that video file access patterns show a news-like lifespan, whereas the audio files follow a regular lifespan. For our log material, it is easy to understand this difference; videos represent news clips and will be consumed as such, whereas the audio files are mostly hitlist songs that will have a longer lifespan but with lower initial access frequency. In our logs, about 70% of the files are video files and thus follow the news-like lifespan, while the remaining 30% are audio files with a regular lifespan.

7. **Accesses**
   Access information is generated using information from all the previous steps.

8. **Synthetic log generation**
   The last step generates the synthetic logs containing all the accesses. These logs can be used for trace driven simulation and evaluation.

In the above discussion, we have presented MediSyn's generation process and related it to our own analysis. It is important to note that each step by itself does not accurately model a media workload; for example the popularity distribution used in step 1 does only provide a long-term distribution of parameters. It is timeless and does not model the popularity correctly for any specific simulated or actual time. A distribution that appropriately models popularity at any given time is only achieved in combination with time-shifted arrival in step 5 and popularity development over time that is added in step 6.

We believe that MediSyn can be used to model NoD workloads, providing some extensions to the generation process are implemented. One issue is that of the bitrate: The bitrate should be configurable and not assumed to be random. Random matching is representative for bitrates in services with many publishers, but for single-publisher systems such as VG the strong correlation between bitrate and file length must be taken into account. Another important issue is that of interactivity: Medisyn does not generate interactivity patterns, but assumes that a user accesses the entire file or a prefix of the file. This may be an adequate model for a VoD system, but the interactivity pattern exhibited by VG users is not that simple. For example, the sudden increase in traffic caused by peaks is not captured by their traffic model. Thus, MediSyn must be expanded to generate more complex content interaction patterns before it can accurately model NoD workloads.

5. Conclusions

In this paper, we have analyzed workload characteristics for a NoD streaming service and discussed the impact these characteristics have on multimedia distribution architecture design. The analysis is based on log material from Norway's largest online newspaper spanning almost two years with a total of 4.6 million accesses. Our most important findings follow.

File access intensity follows a diurnal pattern, which is different from culture to culture. In Norway most accesses, measured both in number of hits and in server bandwidth usage, occur during working hours (showing what people really do while at work). Weekend access patterns differ from the weekday pattern by having fewer hits and a larger variation in access counts. Knowledge of this pattern is important: In a single server system it is neccessary to provision for the highest load, while a CDN can use this knowledge to perform statistical multiplexing of resources.

The Zipf distribution has been used to model file popularity in VoD systems. In this paper, we have shown that it does not accurately model popularity for a NoD service with an open-ended archive. However, it can accurately describe the popularity of those videos that stand for nearly all bandwidth use. The precondition for drawing conclusions from it is a correct use of the stable phase of news' popularity. An accurate popularity model is important for estimating server and network load.

Our study of stream interaction patterns revealed that almost all accesses are for playback, and the majority of accesses start at the beginning of the file. However, most accesses are partial, meaning that the clients do not view the entire file. On average accesses are biased towards the start of the file, with uniformly decreasing interest through the duration of the file. In essence, NoD exhibits low interactivity, and accesses are weighted towards the beginning of the files. This suggests that employing a partial caching scheme may be beneficial in a CDN with NoD workloads.

Some days have an access intensity far surpassing the average. We have investigated how the access frequency increases. We call this peak formation and show that this formation phase can be matched by a hyperbolic tangent function. The availablility of a model implies that a system that can detect that a peak is forming, may be able to reallocate resources to counter the effect. Caching or file replication could be considered as countermeasures. Peak detection is not a trivial task. We leave the evaluation and generalization of a model that can be used for peak modeling and detection for future work.

Synthetic workloads are useful for evaluating system performance. We evaluated the suitability of MediSyn for NoD workload generation, and found it well-suited except for a few shortcomings: The file bitrate should be configurable and not assumed to be random. Also, a NoD-like interactivity model needs to be implemented; MediSyn assumes that all accesses start at the beginning of the file, while we in our analysis observed a significant number of accesses to other parts of the files as well.

References

[1] Computerworld. Nettaviser i toppen (in Norwegian). `http://www.computerworld.no/index.cfm/fuseaction/artikkel/id/50177`. Accessed 2006-09-24.

[2] Frank T. Johnsen, Trude Hafsøe, and Carsten Griwodz. Analysis of Server Workload and Client Interaction in a News-on-Demand Streaming System. In *IEEE ISM*, San Diego, CA, USA, December 2006.

[3] Wenting Tang, Yun Fu, Ludmilla Cherkasova, and Amin Vahdat. Medisyn: A synthetic streaming media service workload generator. In *NOSSDAV*, pages 12–21, Monterey, CA, USA, 2003.

[4] Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caching for Multimedia Streams. In *INFOCOM*, pages 1310–1319, New York, NY, USA, March 1999.

[5] Y.-J. Kim, T. U. Choi, K. O. Jung, Y. K. Kang, S. H. Park, and Ki-Dong Chung. Clustered multi-media NOD: Popularity-based article prefetching and placement. In *IEEE MSS*, pages 194–202, San Diego, CA, USA, 1999.

[6] M. Vilas, X.G. Pañeda, R. García, D. Melendi, and V.G. García. User behaviour analysis of a video-on-demand service with a wide variety of subjects and lengths. In *EUROMICRO*. IEEE Computer Society, 2005.

[7] Xabiel G. Pañeda et al. Study of server workload in large online newspaper. In *MNSA*, Lisbon, Portugal, July 2006.

[8] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large scale video-on-demand systems. In *Proceedings of EuroSys*, Leuven, Belgium, April 2006.

[9] Marcus Rocha, Marcelo Maia, Italo Cunha, Jussara Almeida, and Sérgio Campos. Scalable media streaming to interactive users. *ACM MM November 6-12, 2005, Singapore.*, ACM 1-59593-044-2/05/0011.

[10] A. Dan and D. Sitaram. A generalized interval caching policy for mixed interactive and long video environments. In *MMCN*, San Jose, CA, USA, January 1996.

[11] Ludmila Cherkasova and Minaxi Gupta. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *NOSSDAV*, pages 33–42, Miami, Florida, USA, 2002.

[12] Jussara M. Almeida, Jeffrey Krueger, Derek L. Eager, and Mary K. Vernon. Analysis of educational media server workloads. In *NOSSDAV*, pages 21–30, Port Jefferson, New York, United States, 2001.

[13] Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang. Adaptive and lazy segmentation based proxy caching for streaming media delivery. In *NOSSDAV*, pages 22–31, Monterey, CA, USA, 2003.